Appendix A

PDS Data Object Definitions

This section provides an alphabetical reference of PDS data object definitions, including a description, a list of required and optional keywords, a list of required and optional sub-objects (or child objects), and one or more examples.

NOTE: Any keywords in the Planetary Science Data Dictionary may also be included in the definition of a specific data object definition.

These definitions and examples are provided here for convenience. Additional examples of Data Object Definitions can be obtained by contacting your Data Engineer. As the definitions herein are subject to additions, modifications, and/or refinement, PDS has a web site where the current state of the Data Object Definitions can be ascertained:

http://pdsproto.jpl.nasa.gov/ddcolstdval/newdd/top.cfm

The examples provided in this Appendix have been based on both existing or planned PDS archive products, modified to reflect the most recent version of the PDS standards. They are not intended to represent existing data products and data object definitions designed under previous PDS standards.

The following PDS approved data object definitions are to be used for labeling primary and secondary data objects. For a more detailed discussion on primary and secondary data objects, see the Data Products chapter in this document.

There now exist four new Primitive Data Objects, ARRAY, BIT_ELEMENT (still under review), COLLECTION and ELEMENT. Although these objects are available, they should only be used after careful consideration of the current PDS Data Objects. Please see the PDS Objects chapter in this document for guidelines on the use of primitive objects.

TABLE OF CONTENTS

A.1 Alias....A-3
A.2 Array (Primitive Data Object) A-4
A.3 Bit Column....A-7
A.4 Bit Element (Primitive Data Object)....A-10
A.5 Catalog....A-11
A.6 Collection (Primitive Data Object)....A-14
A.7 Column....A-15
A.8 Container....A-19
A.9 Data Producer....A-25
A.10 Data Supplier A-26
A.11 Directory A-27
A.12 Document....A-29
A.13 Element (Primitive Data Object) A-32
A.14 File A-33
A.15 Gazetteer_Table....A-37
A.16 Header.... A-45
A.17 Histogram....A-47
A.18 History....A-49
A.19 Image....A-52
A.20 Image Map Projection....A-57
A.21 Index_Table A-62
A.22 Palette....A-67
A.23 Qube....A-70
A.24 Series....A-78
A.25 Spectrum A-82
A.26 SPICE Kernel....A-85
A.27 Table A-87
A.28 Text A-107
A.29 Volume....A-109

A.1 ALIAS

The ALIAS object provides a method for identifying alternate terms or names for approved data elements or objects within a data system. The ALIAS object is an optional sub-object of the COLUMN object.

Required Keywords

1. ALIAS_NAME
2. USAGE_NOTE

Optional Keywords

None

Required Objects

None

Optional Objects

None

Example

The following is an example of the usage of the ALIAS object as a subobject of COLUMN in a Magellan ARCDR label:

```
OBJECT                                  = COLUMN
  NAME                                  = ALT_FOOTPRINT_LONGITUDE
  START_BYTE                            = 1
  DATA_TYPE                             = REAL
  BYTES                                 = 10

  OBJECT                                = ALIAS
    ALIAS_NAME                          = AR_LON
    USAGE_NOTE                          = "MAGELLAN MIT ARCDR SIS"
  END_OBJECT                            = ALIAS
END_OBJECT                              = COLUMN
```

A.2 ARRAY (Primitive Data Object)

The ARRAY object is provided to describe dimensioned arrays of homogeneous objects. Note that an ARRAY can contain only a single object, which can itself be another ARRAY or COLLECTION if required. A maximum of 6 axes is allowed in an ARRAY. The optional _AXIS_ elements can be used to describe the variation between successive objects in the ARRAY.

Values for AXIS_ITEMS and _AXIS_ elements for multidimensional arrays are supplied as sequences in which the right most item varies the fastest as the default.

The optional START_BYTE data element provides the starting location relative to an enclosing object. If a START_BYTE is not specified, a value of 1 is assumed.

Required Keywords

1. AXES
2. AXIS_ITEMS
3. NAME

Optional Keywords

1. AXIS_INTERVAL
2. AXIS_NAME
3. AXIS_UNIT
4. AXIS_START
5. AXIS_STOP
6. AXIS_ORDER_TYPE
7. CHECKSUM
8. DESCRIPTION
9. INTERCHANGE_FORMAT
10. START_BYTE

Required Objects

None

Optional Objects

1. ARRAY
2. BIT_ELEMENT
3. COLLECTION
4. ELEMENT

Example 1

The following is an example of a two dimensional Spectrum Array in a detached label.

```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 1600
FILE_RECORDS                    = 180

DATA_SET_ID                     = "IHW-C-SPEC-2-EDR-HALLEY-V1.0"
OBSERVATION_ID                  = "704283"
TARGET_NAME                     = "HALLEY"
INSTRUMENT_HOST_NAME            = "IHW SPECTROSCOPY AND SPECTROPHOTOMETRY NETWORK"
INSTRUMENT_NAME                 = "IHW SPECTROSCOPY AND SPECTROPHOTOMETRY"
PRODUCT_ID                      = "704283"
OBSERVATION_TIME                = 1986-05-09T04:10:20.640Z
START_TIME                      = 1986-05-09T04:07:50.640Z
STOP_TIME                       = UNK
PRODUCT_CREATION_TIME           = 1993-01-01T00:00:00.000Z
^ARRAY                          = "SPEC2702.DAT"
/* Description of Object in File */
OBJECT                          = ARRAY
NAME                            = "2D SPECTRUM"
INTERCHANGE_FORMAT              = BINARY
AXES                            = 2
AXIS_ITEMS                      = (180,800)
AXIS_NAME                       = ("RHO","APPROXIMATE WAVELENGTH")
AXIS_UNIT                       = (ARCSEC,ANGSTROMS)
AXIS_INTERVAL                   = (1.5,7.2164)
AXIS_START                      = (1.0,5034.9)

OBJECT                          = ELEMENT
DATA_TYPE                       = MSB_INTEGER
BYTES                           = 2
NAME                            = COUNT
DERIVED_MAXIMUM                 = 2.424980E+04
DERIVED_MINIMUM                 = 0.000000E+00
OFFSET                          = 0.000000E+00
SCALING_FACTOR                  = 1.000000E+00
NOTE                            = "Conversion factor 1.45 may be applied to data to estimate photons/sq m/sec/
                                   angstrom at 6800 angstroms."
END_OBJECT                      = ELEMENT
END_OBJECT                       = ARRAY
END
```

Example 2

The following is an example of ARRAY, COLLECTION and ELEMENT primitive objects all used together.

```
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 122
FILE_RECORDS                    = 7387

^ARRAY                          = "MISCHA01.DAT"
```

```
DATA_SET_ID                          = "VEGA1-C-MISCHA-3-RDR-HALLEY-V1.0"
TARGET_NAME                          = HALLEY
SPACECRAFT_NAME                      = "VEGA 1"
INSTRUMENT_NAME                      = "MAGNETOMETER"
PRODUCT_ID                           = "XYZ"
START_TIME                           = "UNK"
STOP_TIME                            = "UNK"
SPACECRAFT_CLOCK_START_COUNT         = "UNK"
SPACECRAFT_CLOCK_STOP_COUNT          = "UNK"

NOTE                                 = "VEGA 1 MISCHA DATA"

OBJECT                               = ARRAY
NAME                                 = MISCHA_DATA_FILE
INTERCHANGE_FORMAT                   = BINARY
AXES                                 = 1
AXIS_ITEMS                           = 7387
DESCRIPTION                          = "This file contains an array of fixed length Mischa records."

OBJECT                               = COLLECTION
NAME                                 = MISCHA_RECORD
BYTES                                = 122
DESCRIPTION                          = "Each record in this file consists of a time tag followed by a  20-element array
                                        of magnetic field vectors."

OBJECT                               = ELEMENT
NAME                                 = START_TIME
BYTES                                = 2
DATA_TYPE                            = MSB_INTEGER
START_BYTE                           = 1
END_OBJECT                           = ELEMENT

OBJECT                               = ARRAY
NAME                                 = MAGNETIC_FIELD_ARRAY
AXES                                 = 2
AXIS_ITEMS                           = (3,20)
START_BYTE                           = 3
AXIS_NAME                            = ("XYZ_COMPONENT","TIME"  )
AXIS_UNIT                            = ("N/A"         ,"SECOND")
AXIS_INTERVAL                        = ("N/A"         , 0.2    )
DESCRIPTION                          = "Magnetic field vectors were recorded at the rate
of 10 per second. The START_TIME field gives the time at which the first vector in the record was recorded.  Successive vectors
were recorded  at 0.2 second intervals."

OBJECT                               = ELEMENT
NAME                                 = MAG_FIELD_COMPONENT_VALUE
BYTES                                = 2
DATA_TYPE                            = MSB_INTEGER
START_BYTE                           = 1
END_OBJECT                           = ELEMENT
END_OBJECT                           = ARRAY

END_OBJECT                           = COLLECTION

END_OBJECT                           = ARRAY
END
```

A.3 BIT COLUMN

The BIT_COLUMN object identifies a string of bits that do not fall on even byte boundaries and therefore cannot be described as a distinct COLUMN. BIT_COLUMNS defined within columns are analogous to columns defined within rows.

Note: (1) The Planetary Data System recommends that all fields (within new objects) should be defined on byte boundaries. This precludes having multiple values strung together in bit strings, as occurs in the BIT_COLUMN object.

(2) BIT_COLUMN is intended for use in describing existing binary data strings, but is not recommended for use in defining new data objects because it will not be recognized by most general purpose software.

(3) A BIT_COLUMN must not contain embedded objects.

BIT_COLUMNS of the same format and size may be specified as a single BIT_COLUMN by using the ITEMS, ITEM_BITS, and ITEM_OFFSET elements. The ITEMS data element is used to indicate the number of occurrences of a bit string.

Required Keywords

1. NAME
2. BIT_DATA_TYPE
3. START_BIT
4. BITS (required for BIT_COLUMNs without items)
5. DESCRIPTION

Optional Keywords

1. BIT_MASK
2. BITS (optional for BIT_COLUMNs with items)
3. FORMAT
4. INVALID_CONSTANT
5. ITEMS
6. ITEM_BITS
7. ITEM_OFFSET
8. MINIMUM
9. MAXIMUM
10. MISSING_CONSTANT
11. OFFSET
12. SCALING_FACTOR
13. UNIT

Required Objects

None

Optional Objects

None

Example

The example below was extracted from a larger example which can be found within the CONTAINER object. The BIT_COLUMN object can be a sub-object of the TABLE or CONTAINER object.

```
OBJECT                          =COLUMN
NAME                            =PACKET_ID
DATA_TYPE                       =LSB_BIT_STRING
START_BYTE                      =1
BYTES                           =2
VALID_MINIMUM                   =0
VALID_MAXIMUM                   =7
DESCRIPTION                     = "Packet_id constitutes one of three parts  in the  primary source information
header applied by the Payload Data System (PDS) to the MOLA telemetry packet at the time of creation of the packet prior to
transfer frame creation.  "

OBJECT                          =BIT_COLUMN
NAME                            =VERSION_NUMBER
BIT_DATA_TYPE                   =MSB_UNSIGNED_INTEGER
START_BIT                       =1
BITS                            =3
MINIMUM                         =0
MAXIMUM                         =7
DESCRIPTION                     = "These bits identify Version 1 as the Source Packet structure.  These bits shall
be set to '000'."
END_OBJECT                      =BIT_COLUMN

OBJECT                          =BIT_COLUMN
NAME                            =SPARE
BIT_DATA_TYPE                   =MSB_UNSIGNED_INTEGER
START_BIT                       =4
BITS                            =1
MINIMUM                         =0
MAXIMUM                         =0
DESCRIPTION                     ="Reserved spare.  This bit shall be set to '0'"
END_OBJECT                      =BIT_COLUMN

OBJECT                          =BIT_COLUMN
NAME                            =FLAG
BIT_DATA_TYPE                   =BOOLEAN
START_BIT                       =5
BITS                            =1
MINIMUM                         =0
MAXIMUM                         =0
DESCRIPTION                     ="This flag signals the presence or absence of a Secondary Header data structure
within the Source Packet.   This bit shall be set to '0' since no Secondary Header formatting standards currently exist for Mars
Observer."
END_OBJECT                      =BIT_COLUMN

OBJECT                          =BIT_COLUMN
NAME                            =ERROR_STATUS
```

```
BIT_DATA_TYPE                =MSB_UNSIGNED_INTEGER
START_BIT                    =6
BITS                         =3
MINIMUM                      =0
MAXIMUM                      =7
DESCRIPTION                  ="This field identifies in part the individual application process within the
spacecraft that created the Source Packet data."
END_OBJECT                   = BIT_COLUMN

OBJECT                       = BIT_COLUMN
NAME                         = INSTRUMENT_ID
BIT_DATA_TYPE                = MSB_UNSIGNED_INTEGER
START_BIT                    = 9
BITS                         = 8
MINIMUM                      = "N/A"
MAXIMUM                      = "N/A"
DESCRIPTION                  =  "This field identifies in part the individual application process within the
spacecraft that created the Source Packet data.  00100011 is the bit pattern for MOLA."
END_OBJECT                   = BIT_COLUMN
END_OBJECT                   = COLUMN
```

A.4 BIT ELEMENT (Primitive Data Object)

Under review.

A.5 CATALOG

The CATALOG object is used within a VOLUME object to reference completed PDS high level catalog templates. These templates provide additional information related to the data sets on the volume. Please refer to the File Specification and Naming chapter in this document for more information.

Required Keywords

None

Optional Keywords

1. DATA_SET_ID
2. LOGICAL_VOLUME_PATHNAME
3. LOGICAL_VOLUMES

Required Objects

1. DATA_SET
2. INSTRUMENT
3. INSTRUMENT_HOST
4. MISSION

Optional Objects

1. DATA_SET_COLLECTION
2. PERSONNEL
3. REFERENCE
4. TARGET

Example

The example under the VOLUME object provides an example of a CATALOG object where all the Catalog Templates are included in a single file, CATALOG.CAT.

The example below is a VOLDESC.CAT file that demonstrates multiple data sets per volume. In this example, the Catalog Templates are in separate files and are referenced by the use of pointers. However, the catalog templates may also be included in-line - but this is not the recommended approach (see Section 19.3, PDS Preferred Method for Supplying Catalog Objects).

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
LABEL_REVISION_NOTE             ="RSimpson, 1998-07-01"
RECORD_TYPE                     = STREAM

OBJECT                          = VOLUME
```

```
VOLUME_SERIES_NAME                = "VOYAGERS TO THE OUTER PLANETS"
VOLUME_SET_NAME                   = "VOYAGER NEPTUNE PLANETARY PLASMA INTERACTIONS DATA"
VOLUME_SET_ID                     = USA_NASA_PDS_VG_1001
VOLUMES                           = 1
VOLUME_NAME                       = "VOYAGER NEPTUNE PLANETARY PLASMA INTERACTIONS DATA"
VOLUME_ID                         = VG_1001
VOLUME_VERSION_ID                 = "VERSION 1"
VOLUME_FORMAT                     = "ISO-9660"
MEDIUM_TYPE                       = "CD-ROM"
PUBLICATION_DATE                  = 1992-11-13
DESCRIPTION                       = "This volume contains a collection of non-imaging Planetary Plasma datasets
from the Voyager 2 spacecraft encounter with Neptune.  Included are datasets from the Cosmic Ray System (CRS), Plasma System
(PLS), Plasma Wave System (PWS), Planetary Radio Astronomy (PRA), Magnetometer (MAG), and Low Energy Charged Particle
(LECP) instruments, as well as spacecraft position vectors (POS) in several coordinate systems.  The volume also contains
documentation and index files to support access and use of the data."

DATA_SET_ID                       = { "VG2-N-CRS-3-RDR-D1-6SEC-V1.0",
                                      "VG2-N-CRS-4-SUMM-D1-96SEC-V1.0",
                                      "VG2-N-CRS-4-SUMM-D2-96SEC-V1.0",
                                      "VG2-N-LECP-4-SUMM-SCAN-24SEC-V1.0",
                                      "VG2-N-LECP-4-RDR-STEP-12.8MIN-V1.0",
                                      "VG2-N-MAG-4-RDR-HG-COORDS-1.92SEC-V1.0",
                                      "VG2-N-MAG-4-SUMM-HG-COORDS-48SEC-V1.0",
                                      "VG2-N-MAG-4-RDR-HG-COORDS-9.6SEC-V1.0",
                                      "VG2-N-MAG-4-SUMM-NLSCOORDS-12SEC-V1.0",
                                      "VG2-N-PLS-5-RDR-2PROMAGSPH-48SEC-V1.0",
                                      "VG2-N-PLS-5-RDR-ELEMAGSPHERE-96SEC-V1.0",
                                      "VG2-N-PLS-5-RDR-IONMAGSPHERE-48SEC-V1.0",
                                      "VG2-N-PLS-5-RDR-IONLMODE-48SEC-V1.0",
                                      "VG2-N-PLS-5-RDR-IONMMODE-12MIN-V1.0",
                                      "VG2-N-PLS-5-RDR-ION-INBNDWIND-48SEC-V1.0",
                                      "VG2-N-POS-5-RDR-HGHGCOORDS-48SEC-V1.0",
                                      "VG2-N-POS-5-SUMM-NLSCOORDS-12-48SEC-V1.0",
                                      "VG2-N-PRA-4-SUMM-BROWSE-SEC-V1.0",
                                      "VG2-N-PRA-2-RDR-HIGHRATE-60MS-V1.0",
                                      "VG2-N-PWS-2-RDR-SA-4SEC-V1.0",
                                      "VG2-N-PWS-4-SUMM-SA-48SEC-V1.0",
                                      "VG2-N-PWS-1-EDR-WFRM-60MS-V1.0"}

OBJECT                            = DATA_PRODUCER
INSTITUTION_NAME                  = "UNIVERSITY OF CALIFORNIA, LOS ANGELES"
FACILITY_NAME                     = "PDS PLANETARY PLASMA INTERACTIONS NODE"
FULL_NAME                         = "DR. RAYMOND WALKER"
DISCIPLINE_NAME                   = "PLASMA INTERACTIONS"
ADDRESS_TEXT                      = "UCLA
                                    IGPP
                                    LOS ANGELES, CA 90024  USA"
END_OBJECT                        = DATA_PRODUCER

OBJECT                            = DATA_SUPPLIER
INSTITUTION_NAME                  = "NATIONAL SPACE SCIENCE DATA CENTER"
FACILITY_NAME                     = "NATIONAL SPACE SCIENCE DATA CENTER"
FULL_NAME                         = "NATIONAL SPACE SCIENCE DATA CENTER"
DISCIPLINE_NAME                   = "NATIONAL SPACE SCIENCE DATA CENTER"
ADDRESS_TEXT                      = "Code 633  \n
                                     Goddard Space Flight Center  \n
                                     Greenbelt, Maryland, 20771, USA"
TELEPHONE_NUMBER                  = "3012866695"
ELECTRONIC_MAIL_TYPE              = "NSI/DECNET"
```

```
ELECTRONIC_MAIL_ID                = "NSSDCA::REQUEST"
END_OBJECT                        = DATA_SUPPLIER

OBJECT                            = CATALOG
^MISSION_CATALOG                  = "MISSION.CAT"
^INSTRUMENT_HOST_CATALOG          = "INSTHOST.CAT"
^INSTRUMENT_CATALOG               = {"CRS_INST.CAT",
                                     "LECPINST.CAT",
                                     "MAG_INST.CAT",
                                     "PLS_INST.CAT",
                                     "PRA_INST.CAT",
                                     "PWS_INST.CAT"}
^DATA_SET_CATALOG                 = {"CRS_DS.CAT",
                                     "LECP_DS.CAT",
                                     "MAG_DS.CAT",
                                     "PLS_DS.CAT",
                                     "POS_DS.CAT",
                                     "PRA_DS.CAT",
                                     "PWS_DS.CAT"}
^TARGET_CATALOG                   = TGT.CAT
^PERSONNEL_CATALOG                = PERS.CAT
^REFERENCE_CATALOG                = REFS.CAT
END_OBJECT                        = CATALOG

END_OBJECT                        = VOLUME
END
```

A.6 COLLECTION (Primitive Data Object)

The COLLECTION object allows the ordered grouping of heterogeneous objects into a named collection. The COLLECTION object may contain a mixture of different object types including other COLLECTIONS. The optional START_BYTE data element provides the starting location relative to an enclosing object. If a START_BYTE is not specified, a value of 1 is assumed.

Required Keywords

1. BYTES
2. NAME

Optional Keywords

1. DESCRIPTION
2. CHECKSUM
3. INTERCHANGE_FORMAT
4. START_BYTE

Required Objects

None

Optional Objects

1. ELEMENT
2. BIT_ELEMENT
3. ARRAY
4. COLLECTION

Example

Please refer to the example in the ARRAY Primitive object for an example of an implementation of the COLLECTION object.

A.7 COLUMN

The COLUMN object identifies a single column in a data object.

Note:(1) Current PDS-described data objects that include COLUMN objects are the TABLE, CONTAINER, SPECTRUM and SERIES objects.
(2) COLUMNs must not contain embedded COLUMN objects.
(3) COLUMNs of the same format and size may be specified as a single COLUMN by using the ITEMS, ITEM_BYTES, and ITEM_OFFSET elements. The ITEMS data element indicates the number of occurrences of the field.
(4) BYTES and ITEM_BYTES counts do not include leading or trailing delimiters or line terminators.
(5) For a COLUMN with items, the value of BYTES should represent the size of the column including delimiters between the items. See examples 1 and 2 below.

Required Keywords

1. NAME
2. DATA_TYPE
3. START_BYTE
4. BYTES (required for COLUMNs without items)

Optional Keywords

1. BIT_MASK
2. BYTES (optional for COLUMNs with items)
3. DERIVED_MAXIMUM
4. DERIVED_MINIMUM
5. DESCRIPTION
6 . FORMAT
7. INVALID_CONSTANT
8. ITEM_BYTES
9. ITEM_OFFSET
10. ITEMS
11. MAXIMUM
12. MAXIMUM_SAMPLING_PARAMETER
13. MINIMUM
14. MINIMUM_SAMPLING_PARAMETER
15. MISSING_CONSTANT
16. OFFSET
17. SAMPLING_PARAMETER_INTERVAL
18. SAMPLING_PARAMETER_NAME
19. SAMPLING_PARAMETER_UNIT
20. SCALING_FACTOR
21. UNIT
22. VALID_MAXIMUM
23. VALID_MINIMUM

Required Objects

None

Optional Objects

1. BIT_COLUMN
2. ALIAS

Example 1

The example below shows the use of a COLUMN with items. In this example, the data described is a column with three ASCII_INTEGER items: xx,yy, zz

The ITEM_OFFSET is the number of bytes from the beginning of one item to the beginning of the next.

Note that the value of BYTES includes the comma delimiters between items.

```
OBJECT                          = COLUMN
NAME                            = COLUMNXYZ
DATA_TYPE                       = ASCII_INTEGER
START_BYTE                      = 1
BYTES                           = 8  /*includes delimiters*/
ITEMS                           = 3
ITEM_BYTES                      = 2
ITEM_OFFSET                     = 3
END_OBJECT                      =  COLUMN
```

Example 2

The example below again shows the use of a COLUMN with items. In this example, the data described is a column with three CHARACTER items: "xx", "yy", "zz"

```
OBJECT                          =  COLUMN
NAME                            =  COLUMNXYZ
DATA_TYPE                       =  CHARACTER
START_BYTE                      =  2        /* value does not include leading quote */
BYTES                           =  12       /* value does not include leading and trailing
                                            quotes */
ITEMS                           =  3
ITEM_BYTES                      =  2        /* value does not include leading and trailing
                                            quotes */
ITEM_OFFSET                     =  5        /* value does not include leading quote */
END_OBJECT                      =  COLUMN
```

Example 3

The example below was extracted from a larger example which can be found under the CONTAINER object. The COLUMN object is a sub-object of the TABLE, SERIES, SPECTRUM, and CONTAINER objects.

```
OBJECT                          = COLUMN
NAME                            = PACKET_ID
DATA_TYPE                       = LSB_BIT_STRING
START_BYTE                      = 1
BYTES                           = 2
VALID_MINIMUM                   = 0
VALID_MAXIMUM                   = 7
DESCRIPTION                     = "Packet_id constitutes one of three parts in the  primary source information
header applied by the Payload Data System (PDS) to the MOLA telemetry packet at the time of creation of the packet prior to
transfer frame creation.  "

OBJECT                          = BIT_COLUMN
NAME                            = VERSION_NUMBER
BIT_DATA_TYPE                   = MSB_UNSIGNED_INTEGER
START_BIT                       = 1
BITS                            = 3
MINIMUM                         = 0
MAXIMUM                         = 7
DESCRIPTION                     =  "These bits identify Version 1 as the Source Packet structure.  These bits
shall be set to '000'."
END_OBJECT                      =   BIT_COLUMN

OBJECT                          = BIT_COLUMN
NAME                            = SPARE
BIT_DATA_TYPE                   = MSB_UNSIGNED_INTEGER
START_BIT                       = 4
BITS                            = 1
MINIMUM                         = 0
MAXIMUM                         = 0
DESCRIPTION                     = "Reserved spare.  This bit shall be set to '0"'
END_OBJECT                      =  BIT_COLUMN

OBJECT                          = BIT_COLUMN
NAME                            = FLAG
BIT_DATA_TYPE                   = BOOLEAN
START_BIT                       = 5
BITS                            = 1
MINIMUM                         = 0
MAXIMUM                         = 0
DESCRIPTION                     =  "This flag signals the presence or absence of a Secondary Header data structure
within the Source Packet. This bit shall be set to '0' since no Secondary Header formatting standards currently exist for Mars
Observer."
END_OBJECT                      = BIT_COLUMN

OBJECT                          = BIT_COLUMN
NAME                            = ERROR_STATUS
BIT_DATA_TYPE                   = MSB_UNSIGNED_INTEGER
START_BIT                       = 6
BITS                            = 3
MINIMUM                         = 0
MAXIMUM                         = 7
DESCRIPTION                     = "This field identifies in part the individual application process within the
```

spacecraft that created the Source Packet data."
END_OBJECT = BIT_COLUMN

OBJECT = BIT_COLUMN
NAME = INSTRUMENT_ID
BIT_DATA_TYPE = MSB_UNSIGNED_INTEGER
START_BIT = 9
BITS = 8
MINIMUM = "N/A"
MAXIMUM = "N/A"
DESCRIPTION = "This field identifies in part the individual application process within the spacecraft that creeated the Source Packet data. 00100011 is the bit pattern for MOLA."
END_OBJECT = BIT_COLUMN
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = CH_4_2ND_HALF_FRAME_BKGRND_CN
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 134
BYTES = 1
MINIMUM = 0
MAXIMUM = 255
DESCRIPTION = "The background energy or noise count levels in channels 1, 2, 3, and 4 respectively by half-frame. Pseudo log value of NOISE(1, 2, 3, 4) at the end of a half-frame of current frame, 5.3 bit format. Plog base 2 of background count sum..."
END_OBJECT = COLUMN

A.8 CONTAINER

The CONTAINER object is used to group a set of sub-objects (such as COLUMNS) that repeat within a data object (such as a TABLE). Use of the CONTAINER object allows repeating groups to be defined within a data structure.

Required Keywords

1. NAME
2. START_BYTE
3. BYTES
4. REPETITIONS
5. DESCRIPTION

Optional Keywords

None

Required Objects

None

Optional Objects

1. COLUMN
2. CONTAINER

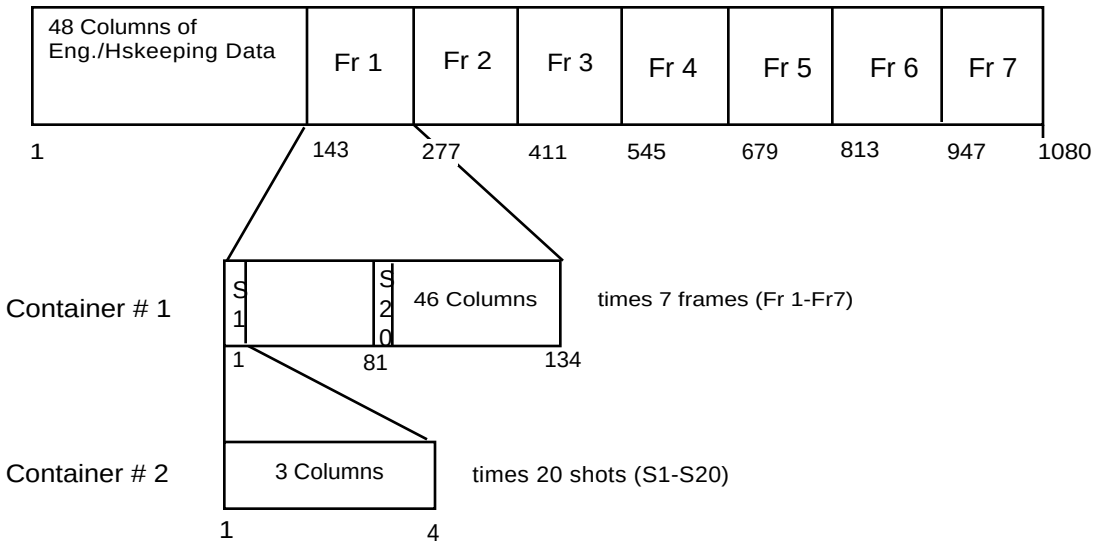
Example

The following diagram shows a data product layout in which the CONTAINER object is used. The diagram depicts the modelled data product as a TABLE with one row (or one record of data). Each record within the diagram begins with 48 columns (143 bytes) of engineering data. The data product acquires science data from seven different frames. Since the data from each frame are formatted identically, one CONTAINER description can suffice for all seven frames.

In this example there are two CONTAINER objects. The first CONTAINER object describes the repeating frame information. Within this CONTAINER there is a second CONTAINER object in which a 4-byte set of three COLUMN objects repeats 20 times. The use of the second CONTAINER object permits the data supplier to describe the three COLUMNS (4 bytes) once, instead of specifying sixty column definitions.

In the first CONTAINER, the keyword REPETITIONS is equal to 7. In the second CONTAINER, REPETITIONS equals 20. Both CONTAINER objects contain a collection of COLUMN objects. In most cases it is preferable to save space in the product label by placing COLUMN objects in a separate file and pointing to that file from within the CONTAINER object.

.



This attached label example describes the above TABLE structure using CONTAINER objects.

CCSD3ZF0000100000001NJPL3KS0PDSXAAAAAAAA
PDS_VERSION_ID = PDS3
RECORD_TYPE = FIXED_LENGTH
FILE_RECORDS = 467
RECORD_BYTES = 1080
LABEL_RECORDS = 4
FILE_NAME = "AEDR.001"

^MOLA_SCIENCE_MODE_TABLE = 5
DATA_SET_ID = "MO-M-MOLA-1-AEDR-L0-V1.0"
PRODUCT_ID = "MOLA-AEDR.-10010-0001"
SPACECRAFT_NAME = MARS_OBSERVER
INSTRUMENT_ID = MOLA
INSTRUMENT_NAME = MARS_OBSERVER_LASER_ALTIMETER
TARGET_NAME = MARS
SOFTWARE_NAME = "Browser 17.1"
UPLOAD_ID = "5.3"
PRODUCT_RELEASE_DATE = 1994-12-29T02:10:09.321
START_TIME = 1994-09-29T04:12:43.983
STOP_TIME = 1994-09-29T06:09:54.221
SPACECRAFT_CLOCK_START_COUNT = "12345"
SPACECRAFT_CLOCK_STOP_COUNT = "12447"
PRODUCT_CREATION_TIME = 1995-01-29T07:30:333
MISSION_PHASE_NAME = MAPPING
ORBIT_NUMBER = 0001
PRODUCER_ID = MO_MOLA_TEAM
PRODUCER_FULL_NAME = "DAVID E. SMITH"
PRODUCER_INSTITUTION_NAME = "GODDARD SPACE FLIGHT CENTER"
DESCRIPTION = "This data product contains the aggregation of MOLA telemetry packets by Orbit. All Experiment Data Record Packets retrieved from the PDB are collected in this data product. The AEDR data product is put together with the Project-provided software tool Browser."

```
OBJECT                                  =  MOLA_SCIENCE_MODE_TABLE
INTERCHANGE_FORMAT                      =  BINARY
ROWS                                    =  463
COLUMNS                                 =  97
ROW_BYTES                               =  1080
^STRUCTURE                              =  "MOLASCI.FMT"
DESCRIPTION                             =  "This table is one of two that describe the arrangement of information on the
Mars Observer Laser Altimeter (MOLA) Aggregated Engineering Data Record (AEDR).  ..."

END_OBJECT                              =     MOLA_SCIENCE_MODE_TABLE
...

END
CCSD$$MARK$$AAAAAAAANJPL3IF0NNNN00000001
```

Contents of the MOLASCI.FMT file

```
OBJECT                                  =  COLUMN
NAME                                    =  PACKET_ID
DATA_TYPE                               =  LSB_BIT_STRING
START_BYTE                              =  1
BYTES                                   =  2
VALID_MINIMUM                           =  0
VALID_MAXIMUM                           =  7
DESCRIPTION                             =  "Packet_id constitutes one of three parts in the  primary source information
header applied by the Payload Data System (PDS) to the MOLA telemetry packet at the time of creation of the packet prior to
transfer frame creation.  "

OBJECT                                  =  BIT_COLUMN
NAME                                    =  VERSION_NUMBER
BIT_DATA_TYPE                           =  UNSIGNED_INTEGER
START_BIT                               =  1
BITS                                    =  3
MINIMUM                                 =  0
MAXIMUM                                 =  7
DESCRIPTION                             =   "These bits identify Version 1 as the Source Packet structure.  These bits shall
be set to '000'."
END_OBJECT                              =  BIT_COLUMN

OBJECT                                  =  BIT_COLUMN
NAME                                    =  SPARE
BIT_DATA_TYPE                           =  UNSIGNED_INTEGER
START_BIT                               =  4
BITS                                    =  1
MINIMUM                                 =  0
MAXIMUM                                 =  0
DESCRIPTION                             =  "Reserved spare.  This bit shall be set to '0'"
END_OBJECT                              =  BIT_COLUMN

OBJECT                                  =  BIT_COLUMN
NAME                                    =  SECONDARY_HEADER_FLAG
BIT_DATA_TYPE                           =  BOOLEAN
START_BIT                               =  5
BITS                                    =  1
MINIMUM                                 =  0
MAXIMUM                                 =  0
DESCRIPTION                             =  "This flag signals the presence or absence of a Secondary Header data
structure within the Source Packet. This bit shall be set to '0' since no Secondary Header formatting standards currently exist for
```

```
Mars Observer."
END_OBJECT                              =  BIT_COLUMN

OBJECT                                  =  BIT_COLUMN
NAME                                    =  ERROR_STATUS
BIT_DATA_TYPE                           =  UNSIGNED_INTEGER
START_BIT                               =   6
BITS                                    =  3
MINIMUM                                 =  0
MAXIMUM                                 =  7
DESCRIPTION                             =   "This field identifies in part the individual application process within the
spacecraft that created the Source Packet data."
END_OBJECT                              =  BIT_COLUMN

OBJECT                                  =  BIT_COLUMN
NAME                                    =  INSTRUMENT_ID
BIT_DATA_TYPE                           =  UNSIGNED_INTEGER
START_BIT                               =  9
BITS                                    =  8
MINIMUM                                 =  2#0100011#
MAXIMUM                                 =  2#0100011#
DESCRIPTION                             =    "This field identifies in part the individual application process within the
spacecraft that created the Source Packet data.  00100011 is the bit pattern for MOLA."
END_OBJECT                              =  BIT_COLUMN
END_OBJECT                              =  COLUMN

...

OBJECT                                  =  COLUMN
NAME                                    =  COMMAND_ECHO
DATA_TYPE                               =  INTEGER
START_BYTE                              =  125
BYTES                                   =  16
ITEMS                                   =  8
ITEM_BYTES                              =  2
MINIMUM                                 =  0
MAXIMUM                                 =  65535
DESCRIPTION                             =  "First 8 command words received during current packet, only complete
commands are stored, MOLA specific commands only. The software attempts to echo all valid commands.  If the command will fit
in the room remaining in the..."
END_OBJECT                              =  COLUMN

OBJECT                                  =  COLUMN
NAME                                    =  PACKET_VALIDITY_CHECKSUM
DATA TYPE                               =  INTEGER
START_BYTE                              =  141
BYTES                                   =  2
MINIMUM                                 =  0
MAXIMUM                                 =  65535
DESCRIPTION                             =   "Simple 16 bit addition of entire packet contents upon completion.  This
location is zeroed for addition. This word is zeroed, then words 0-539 are added without carry to a variable that is initially zero. The
resulting lower 16 bits ar..."
END_OBJECT                              =  COLUMN

OBJECT                                  =  CONTAINER
NAME                                    =  FRAME_STRUCTURE
^STRUCTURE                              =   "MOLASCFR.FMT"  /*points to the columns */
 /*that make up the frame descriptors */
START_BYTE                              =  143
```

BYTES = 134
REPETITIONS = 7
DESCRIPTION = "The frame_structure container represents the format of seven repeating groups of attributes in this data product. The data product reflects science data acquisition from seven different frames. Since the data from each frame are ..."
END_OBJECT = CONTAINER

CONTENTS OF THE MOLASCFR.FMT FILE

OBJECT = CONTAINER
NAME = COUNTS
START_BYTE = 1
BYTES = 4
REPETITIONS = 20
^STRUCTURE = "MOLASCCT.FMT"
DESCRIPTION = "This container has three sub-elements (range to surface counts, 1st channel received pulse energy, and 2nd channel received pulse energy). The three sub-elements repeat for each of 20 shots."
END_OBJECT = CONTAINER

OBJECT = COLUMN
NAME = SHOT_2_LASER_TRANSMITTER_POWR
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 81
BYTES = 1
MINIMUM = 0
MAXIMUM = 65535
DESCRIPTION = "..."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SHOT_1_LASER_TRANSMITTER_POWR
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 82
BYTES = 1
MINIMUM = 0
MAXIMUM = 65535
DESCRIPTION = "..."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SHOT_4_LASER_TRANSMITTER_POWR
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 83
BYTES = 1
MINIMUM = 0
MAXIMUM = 65535
DESCRIPTION = "..."
END_OBJECT = COLUMN

...

OBJECT = COLUMN
NAME = CH_3_2ND_HALF_FRAME_BKGRND_CN
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 133
BYTES = 1
MINIMUM = 0
MAXIMUM = 255

DESCRIPTION = "The background energy or noise count levels in channels 1, 2, 3, and 4 respectively by half-frame. Pseudo log value of NOISE(1, 2, 3, 4) at the end of a half-frame of current frame, 5.3 bit format. Plog base 2 of background count sum..."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = CH_4_2ND_HALF_FRAME_BKGRND_CN
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 134
BYTES = 1
MINIMUM = 0
MAXIMUM = 255
DESCRIPTION = "The background energy or noise count levels in channels 1, 2, 3, and 4 respectively by half-frame. Pseudo log value of NOISE(1, 2, 3, 4) at the end of a half-frame of current frame, 5.3 bit format. Plog base 2 of background count sum..."
END_OBJECT = COLUMN

CONTENTS OF THE MOLASCCT.FMT FILE

OBJECT = COLUMN
NAME = RANGE_TO_SURFACE_TIU_CNTS
DATA_TYPE = MSB_INTEGER
START_BYTE = 1
BYTES = 2
DESCRIPTION = "The possible 20 valid frame laser shots surface ranging measurements in Timing Interval Unit (TIU) counts. The least significant 16 bits of TIU (SLTIU), stored for every shot. B[0] = Bits 15-8 of TIU reading; B[1] = Bits 7-0 of ..."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = FIRST_CH_RCVD_PULSE_ENRGY
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 3
BYTES = 1
DESCRIPTION = "The level of return, reflected energy as received by the first channel and matched filter to trigger. This is a set of values for all possible 20 shots within the frame. Lowest numbered non-zero energy reading for each shot."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SECOND_CH_RCVD_PULSE_ENRGY
DATA_TYPE = UNSIGNED_INTEGER
START_BYTE = 4
BYTES = 1
DESCRIPTION = "The level of return, reflected energy as received by the second channel and matched filter to trigger. This is a set of values for all possible 20 shots within the frame. 2nd lowest numbered non-zero energy reading for each shot..."
END_OBJECT = COLUMN

A.9 DATA PRODUCER

The DATA_PRODUCER object is used within a PDS object, such as VOLUME. The DATA_PRODUCER, as opposed to the DATA_SUPPLIER, is an individual or organization responsible for collecting, assembling, and/or engineering the raw data into one or more data sets.

Required Keywords

1. INSTITUTION_NAME
2. FACILITY_NAME
3. FULL_NAME
4. ADDRESS_TEXT

Optional Keywords

1. DISCIPLINE_NAME
2. NODE_NAME
3. TELEPHONE_NUMBER
4. ELECTRONIC_MAIL_TYPE
5. ELECTRONIC_MAIL_ID

Required Objects

None

Optional Objects

None

Example

The example below was extracted from a larger example which can be found within the VOLUME object. The DATA_PRODUCER object is a required object of the VOLUME.

```
OBJECT                          = DATA_PRODUCER
INSTITUTION_NAME                = "U.S.G.S. FLAGSTAFF"
FACILITY_NAME                   = "BRANCH OF ASTROGEOLOGY"
FULL_NAME                       = "Eric M. Eliason"
DISCIPLINE_NAME                 = "IMAGE PROCESSING"
ADDRESS_TEXT                    = " Branch of Astrogeology
                                  United States Geological Survey
                                  2255 North Gemini Drive
                                  Flagstaff, Arizona. 86001 USA"
END_OBJECT                      = DATA_PRODUCER
```

A.10 DATA SUPPLIER

The DATA_SUPPLIER object is used within a PDS object, such as VOLUME. The DATA_SUPPLIER, as opposed to the DATA_PRODUCER, is an individual or organization responsible for distributing the data sets and associated data to the science community.

Required Keywords

1. INSTITUTION_NAME
2. FACILITY_NAME
3. FULL_NAME
4. ADDRESS_TEXT
5. TELEPHONE_NUMBER
6. ELECTRONIC_MAIL_TYPE
7. ELECTRONIC_MAIL_ID

Optional Keywords

1. DISCIPLINE_NAME
2. NODE_NAME

Required Objects

None

Optional Objects

None

Example

The example below was extracted from a larger example which can be found within the VOLUME object. The DATA_SUPPLIER object is an optional object of the VOLUME.

```
OBJECT                          = DATA_SUPPLIER
INSTITUTION_NAME                = "NATIONAL SPACE SCIENCE DATA CENTER"
FACILITY_NAME                   = "NATIONAL SPACE SCIENCE DATA CENTER"
FULL_NAME                       = "NATIONAL SPACE SCIENCE DATA CENTER"
DISCIPLINE_NAME                 = "NATIONAL SPACE SCIENCE DATA CENTER"
ADDRESS_TEXT                    = "Code 633
                                  Goddard Space Flight Center
                                  Greenbelt, Maryland, 20771, USA"
 TELEPHONE_NUMBER               = "3012866695"
ELECTRONIC_MAIL_TYPE            = "NSI/DECNET"
ELECTRONIC_MAIL_ID              = "NSSDCA::REQUEST"
END_OBJECT                      = DATA_SUPPLIER
```

A.11 DIRECTORY

The DIRECTORY object is used to define a hierarchical file organization on a linear (sequential) media, such as tape. The DIRECTORY object identifies all directories and subdirectories below the root level, and is a required sub-object of the VOLUME object for tape media.

Note: The root directory on a volume does not need to be explicitly defined with the DIRECTORY object.

Subdirectories are identified by embedding DIRECTORY objects. Files within the directories and subdirectories are sequentially identified by using FILE objects with a sequence_number value corresponding to their position on the media. A sequence_number value will be unique for each file on the media. This format is strongly recommended when transferring or archiving volumes of data on media which do not support hierarchical directory structures (i.e., submitting a tape volume of data for pre-mastering or preparing an archive tape).

Although the DIRECTORY object is optional in the VOLUME object, it is a required object for tape media.

Required Keywords

1. NAME

Optional Keywords

1. RECORD_TYPE
2. SEQUENCE_NUMBER

Required Objects

1. FILE

Optional Objects

1. DIRECTORY

Example

The example below was extracted from a larger example which can be found within the VOLUME object.

```
OBJECT                          = DIRECTORY
NAME                            = INDEX

OBJECT                          = FILE
FILE_NAME                       = "INDXINFO.TXT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 5
END_OBJECT                      = FILE

OBJECT                          = FILE
FILE_NAME                       = "INDEX.LBL"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 6
END_OBJECT                      = FILE

OBJECT                          = FILE
FILE_NAME                       = "INDEX.TAB"
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 512
FILE_RECORDS                    = 6822
SEQUENCE_NUMBER                 = 7
END_OBJECT                      = FILE
END_OBJECT                      = DIRECTORY
```

A.12 DOCUMENT

The DOCUMENT object is used to label a particular document that is provided on a volume to support an archived data product. A document can be made up of one or more files in a single format. For instance, a document may be comprised of as many TIFF files as there are pages in the document.

Multiple versions of a document can be supplied on a volume with separate formats, requiring a DOCUMENT object for each document version (i.e., OBJECT = TEX_DOCUMENT and OBJECT = PS_DOCUMENT when including both the TEX and Postscript versions of the same document).

PDS requires that at least one version of any document be plain ASCII text in order to allow users the capability to read, browse, or search the text without requiring software or text processing packages. This version can be plain, unmarked text, or ASCII text containing a markup language. (See the Documentation chapter of this document for more details.)

The DOCUMENT object contains keywords that identify and describe the document, provide the date of publication of the document, indicate the number of files comprising the document, provide the format of the document files, and identify the software used to compress or encode the document, as applicable.

DOCUMENT labels must be detached files unless the files are plain, unmarked text that will not be read by text or word processing packages. A DOCUMENT object for each format type of a document can be included in the same label file with pointers, such as ^TIFF_DOCUMENT for a TIFF formatted document. (See example below.)

Required Keywords

1. DOCUMENT_NAME
2. DOCUMENT_TOPIC_TYPE
3. INTERCHANGE_FORMAT
4. DOCUMENT_FORMAT
5. PUBLICATION_DATE

Optional Keywords

1. ABSTRACT_TEXT
2. DESCRIPTION
3. ENCODING_TYPE
4. FILES

Required Objects

None

Optional Objects

None

Example

The following example detached label, PDSUG.LBL, is for a Document provided in three formats: ASCII text, TIFF, and TEX.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = UNDEFINED

^ASCII_DOCUMENT                 = "PDSUG.ASC"
^TIFF_DOCUMENT                  = { "PDSUG001.TIF", "PDSUG002.TIF",
                                    "PDSUG003.TIF", "PDSUG004.TIF" }
^TEX_DOCUMENT                   = "PDSUG.TEX"

OBJECT                          = ASCII_DOCUMENT
DOCUMENT_NAME                   = "Planetary Data System Data Set Catalog User's Guide"
PUBLICATION_DATE                = 1992-04-13
DOCUMENT_TOPIC_TYPE             = "USER'S GUIDE"
INTERCHANGE_FORMAT              = ASCII
DOCUMENT_FORMAT                 = TEXT

DESCRIPTION                     = "The Planetary Data System Data Set Catalog User's Guide describes the
fundamentals of accessing, searching, browsing, and ordering data from the PDS Data Set Catalog at the Central Node.  The text
for this 4-page document is provided here in this plain, ASCII text file."
ABSTRACT_TEXT                   = "The PDS Data Set Catalog is similar in function and purpose to a card catalog
in a library.  Use a Search screen to find data items, a List/Order screen to order data items, and the More menu option to see more
information."
END_OBJECT                      = ASCII_DOCUMENT

OBJECT                          = TIFF_DOCUMENT
DOCUMENT_NAME                   = "Planetary Data System Data Set Catalog User's Guide"
DOCUMENT_TOPIC_TYPE             = "USER'S GUIDE"
INTERCHANGE_FORMAT              =  BINARY
DOCUMENT_FORMAT                 =  TIFF
PUBLICATION_DATE                = 1992-04-13
FILES                           = 4
ENCODING_TYPE                   =  "CCITT/3"
DESCRIPTION                     = "
The Planetary Data System Data Set Catalog User's Guide describes the fundamentals of accessing, searching, browsing, and
ordering data from the PDS Data Set Catalog at the Central Node.
The 4-page document is provided here in 4 consecutive files, one file per page, in Tagged Image File Format (TIFF) using Group
3 compression.  It has been tested to successfully import into WordPerfect 5.0, FrameMaker, and Photoshop."
ABSTRACT_TEXT                   = "
The PDS Data Set Catalog is similar in function and purpose to a card catalog in a library.  Use a Search screen to find data items,
a List/Order screen to order data items, and the More menu option to see more information."
END_OBJECT                      = TIFF_DOCUMENT

OBJECT                          = TEX_DOCUMENT
DOCUMENT_NAME                   = "Planetary Data System Data Set Catalog User's Guide"
DOCUMENT_TOPIC_TYPE             = "USER'S GUIDE"
```

```
INTERCHANGE_FORMAT              =  ASCII
DOCUMENT_FORMAT                 =  TEX
PUBLICATION_DATE                = 1992-04-13
DESCRIPTION                     = "
The Planetary Data System Data Set Catalog User's Guide describes the fundamentals of accessing, searching, browsing, and
ordering data from the PDS Data Set Catalog at the Central Node.
The 4-page document is provided here in TeX format with all necessary macros included."
ABSTRACT_TEXT                   = "
The PDS Data Set Catalog is similar in function and purpose to a card catalog in a library.  Use a Search screen to find data items,
a List/Order screen to order data items, and the More menu option to see more information."
END_OBJECT                      = TEX_DOCUMENT
END
```

A.13 ELEMENT (Primitive Data Object)

The ELEMENT object provides a means of defining a lowest level component of a data object that is stored in an integral multiple of 8-bit bytes. Element objects may be embedded in COLLECTION and ARRAY data objects. The optional START_BYTE element identifies a location relative to the enclosing object. If not explicitly included, a START_BYTE = 1 is assumed for the ELEMENT.

Required Keywords

1. BYTES
2. DATA_TYPE
3. NAME

Optional Keywords

1. START_BYTE
2. BIT_MASK
3. DERIVED_MAXIMUM
4. DERIVED_MINIMUM
5. DESCRIPTION
6. FORMAT
7. INVALID_CONSTANT
8. MINIMUM
9. MAXIMUM
10. MISSING_CONSTANT
11. OFFSET
12. SCALING_FACTOR
13. UNIT
14. VALID_MINIMUM
15. VALID_MAXIMUM

Required Objects

None

Optional Objects

None

Example

Please refer to the example in the ARRAY Primitive object for an example of the implementation of the ELEMENT object.

A.14 FILE

The FILE object is used in attached or detached labels to define the attributes or characteristics of a data file. In attached labels, the file object is also used to indicate boundaries between label records and data records in data files which have attached labels. The FILE object may be used in three ways:

(1) As an implicit object in attached or detached labels. As depicted in the following example, all detached label files and attached labels contain an implicit FILE object which starts at the top of the label and ends where the label ends. In these cases, the PDS recommends against using the NAME keyword to reference the file name.

```
---------------------------------------------------------------------
        RECORD_TYPE       = FIXED_LENGTH
        RECORD_BYTES      = 80
        FILE_RECORDS      = 522
        LABEL_RECORDS     = 10
       (remainder of the label)
---------------------------------------------------------------------
```

For data products labelled using the implicit file object (e.g. for minimal labels) DATA_OBJECT_TYPE = FILE should be used in the Data Set Catalog Template.

(2) As an explicit object which is used when a file reference is needed in a combined detached or minimal label. In this case, the optional FILE_NAME element is used to identify the file being referenced.

```
---------------------------------------------------------------------
        OBJECT            = FILE
        FILE_NAME         = "IM10347.DAT"
        RECORD_TYPE       = STREAM
        FILE_RECORDS      = 1024
     (other optional keywords describing the file)
        END_OBJECT        = FILE
---------------------------------------------------------------------
```

For data products labelled using the explicit file object (e.g. for minimal labels) DATA_OBJECT_TYPE = FILE should be used in the Data Set Catalog Template.

(3) As an explicit object to identify specific files as sub-objects of the DIRECTORY in VOLUME objects. In this case, the optional FILE_NAME element is used to identify the file being referenced on a tape archive volume.

```
 ---------------------------------------------------------------------
        OBJECT            = FILE
        FILE_NAME         = "VOLDESC.CAT"
        RECORD_TYPE       = STREAM
        SEQUENCE_NUMBER = 1
        END_OBJECT        = FILE
---------------------------------------------------------------------
```

The keywords in the FILE object always describe the file being referenced, and not the file in which the keywords are contained (i.e., if the FILE object is used in a detached label file, the FILE object keywords describe the detached data file, not the label file which contains the keywords). For example, if a detached label for a data file is being created and the label will be in STREAM format, but the data will be stored in a file having FIXED_LENGTH records, then the RECORD_TYPE keyword in the label file must be given the value FIXED_LENGTH.

The following table identifies data elements that are required (Req), optional (Opt), and not applicable (-) for various types of files

Labeling Method	Att	Det	Att	Det	Att	Det	Att	Det
RECORD_TYPE	FIXED_LENGTH		VARIABLE_LENGTH		STREAM		UNDEFINED	
RECORD_BYTES	Req	Req	Rmax	Rmax	Omax	-	-	-
FILE_RECORDS	Req	Req	Req	Req	Opt	Opt	-	-
LABEL_RECORDS	Req	-	Req	-	Opt	-	-	-

Required Keywords

1. RECORD_TYPE

(See above table for the conditions of use of additional required keywords)

Optional Keywords

1. FILE_NAME (required only in minimal detached labels and tape archives)
2. FILE_RECORDS (required only in minimal detached labels and tape archives)
3. LABEL_RECORDS
4. RECORD_BYTES
5. SEQUENCE_NUMBER

Required Objects

None

Optional Objects

None

Example

Below is an example of a set of explicit file objects in a combined detached label. An additional example of the use of explicit FILE object can be found in the VOLUME object.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
HARDWARE_MODEL_ID               = "SUN SPARC STATION"
OPERATING_SYSTEM_ID             = "SUN OS 4.1.1"
SPACECRAFT_NAME                 = "VOYAGER 2"
INSTRUMENT_NAME                 = "PLASMA WAVE RECEIVER"
MISSION_PHASE_NAME              = "URANUS ENCOUNTER"
TARGET_NAME                     = URANUS
DATA_SET_ID                     = "VG2-U-PWS-4-RDR-SA-48.0SEC-V1.0"
PRODUCT_ID                      = "T860123-T860125"

OBJECT                          = FILE
FILE_NAME                       = "T860123.DAT"
FILE_RECORDS                    = 1800
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 105
START_TIME                      = 1986-01-23T00:00:00.000Z
STOP_TIME                       = 1986-01-24T00:00:00.000Z
^TIME_SERIES                    = "T860123.DAT"

OBJECT                          = TIME_SERIES
INTERCHANGE_FORMAT              = BINARY
ROWS                            = 1800
ROW_BYTES                       = 105
COLUMNS                         = 19
^STRUCTURE                      = "PWS_DATA.FMT"
SAMPLING_PARAMETER_NAME         = TIME
SAMPLING_PARAMETER_UNIT         = SECOND
SAMPLING_PARAMETER_INTERVAL     = 48.0
END_OBJECT                      = TIME_SERIES
END_OBJECT                      = FILE

OBJECT                          = FILE
FILE_NAME                       = "T860124.DAT"
FILE_RECORDS                    = 1800
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 105
START_TIME                      = 1986-01-24T00:00:00.000Z
STOP_TIME                       = 1986-01-25T00:00:00.000Z
^TIME_SERIES                    = "T860124.DAT"

OBJECT                          = TIME_SERIES
INTERCHANGE_FORMAT              = BINARY
ROWS                            = 1800
ROW_BYTES                       = 105
COLUMNS                         = 19
^STRUCTURE                      = "PWS_DATA.FMT"
SAMPLING_PARAMETER_NAME         = TIME
SAMPLING_PARAMETER_UNIT         = SECOND
SAMPLING_PARAMETER_INTERVAL     = 48.0
END_OBJECT                      = TIME_SERIES
END_OBJECT                      = FILE
```

```
OBJECT                          = FILE
FILE_NAME                       = "T860125.DAT"
FILE_RECORDS                    = 1799
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 105
START_TIME                      = 1986-01-30T00:00:00.000Z
STOP_TIME                       = 1986-01-30T23:59:12.000Z
^TIME_SERIES                    = "T860125.DAT"

OBJECT                          = TIME_SERIES
INTERCHANGE_FORMAT              = BINARY
ROWS                            = 1799
ROW_BYTES                       = 105
COLUMNS                         = 19
^STRUCTURE                      = "PWS_DATA.FMT"
SAMPLING_PARAMETER_NAME         = TIME
SAMPLING_PARAMETER_UNIT         = SECOND
SAMPLING_PARAMETER_INTERVAL     = 48.0
END_OBJECT                      = TIME_SERIES
END_OBJECT                      = FILE

END
```

A.15 GAZETTEER_TABLE

The GAZETTEER_TABLE object is a specific type of a TABLE object that provides information about the geographical features of a planet or satellite. It contains information about a named feature such as location, size, origin of feature name, etc. The GAZETTEER_TABLE contains one row for each feature named on the target body. The table is formatted so that it may be read directly by many data management systems on various host computers. All fields (columns) are separated by commas, and character fields are enclosed by double quotation marks. Each record consist of 480 bytes, with a carriage return/line feed sequence in bytes 479 and 480. This allows the table to be treated as a fixed length record file on hosts that support this file type and as a normal text file on other hosts.

Currently the PDS Imaging Node at the USGS is the data producer for all GAZETTEER_TABLEs.

Required Keywords

1. NAME
2. INTERCHANGE_FORMAT
3. ROWS
4. COLUMNS
5. ROW_BYTES
6. DESCRIPTION

Optional Keywords

None

Required Objects

1. COLUMN

Optional Objects

None

Required COLUMN Objects (NAME =)

TARGET_NAME
SEARCH_FEATURE_NAME
DIACRITIC_FEATURE_NAME
MINIMUM_LATITUDE
MAXIMUM_LATITUDE
CENTER_LATITUDE
MINIMUM_LONGITUDE
MAXIMUM_LONGITUDE

CENTER_LONGITUDE
LABEL_POSITION_ID
FEATURE_LENGTH
PRIMARY_PARENTAGE_ID
SECONDARY_PARENTAGE_ID
MAP_SERIAL_ID
FEATURE_STATUS_TYPE
APPROVAL_DATE
FEATURE_TYPE
REFERENCE_NUMBER
MAP_CHART_ID
FEATURE_DESCRIPTION

Required Keywords (for Required COLUMN Objects)

NAME
DATA_TYPE
START_BYTE
BYTES
FORMAT
UNIT
DESCRIPTION

Example

CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID = PDS3
RECORD_TYPE = FIXED_LENGTH
RECORD_BYTES = 480
FILE_RECORDS = 1181
PRODUCT_ID =XYZ
TARGET_NAME = MARS
^GAZETTEER_TABLE = "GAZETTER.TAB"

OBJECT = GAZETTEER_TABLE
NAME = "PLANETARY NOMENCLATURE GAZETTEER"
INTERCHANGE_FORMAT = ASCII
ROWS = 1181
COLUMNS = 20
ROW_BYTES = 480
DESCRIPTION = "The gazetteer (file: GAZETTER.TAB) is a table of geographical features for a planet or satellite. It contains information about a named feature such as location, size, origin of feature name, etc. The Gazetteer Table contains one row for each feature named on the target body. The table is formatted so that it may be read directly into many data management systems on various host computers. All fields (columns) are separated by commas, and character fields are preceded by double quotation marks. Each record consist of 480 bytes, with a carriage return/line feed sequence in bytes 479 and 480. This allows the table to be treated as a fixed length record file on hosts that support this file type and as a normal text file on other hosts."

OBJECT = COLUMN
NAME = TARGET_NAME
DATA_TYPE = CHARACTER

START_BYTE = 2
BYTES = 20
FORMAT = "A20"
UNIT = "N/A"
DESCRIPTION = "The planet or satellite on which the feature is located."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SEARCH_FEATURE_NAME
DATA_TYPE = CHARACTER
START_BYTE = 25
BYTES = 50
FORMAT = "A50"
UNIT = "N/A"
DESCRIPTION = "The geographical feature name with all diacritical marks stripped off. This name is stored in upper case only so that it can be used for sorting and search purposes. This field should not be used to designate the name of the feature because it does not contain the diacritical marks. Feature names not containing diacritical marks can often take on a completely different meaning and in some cases the meaning can be deeply offensive."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = DIACRITIC_FEATURE_NAME
DATA_TYPE = CHARACTER
START_BYTE = 78
BYTES = 100
FORMAT = "A100"
UNIT = "N/A"
DESCRIPTION = "The geographical feature name containing standard diacritical information. A detailed description of the diacritical mark formats are described in the gazetteer documentation.

DIACRITICALS USED IN THE TABLE

The word diacritic comes from a Greek word meaning to separate. It refers to the accent marks employed to separate, or distinguish, one form of pronunciation of a vowel or consonant from another.

This note is included to familiarize the user with the codes used to represent diacriticals found in the table, and the values usually associated with them. In the table, the code for a diacritical is preceded by a backslash and is followed, without a space, by the letter it is modifying.

This note is organized as follows: the code is listed first, followed by the name of the accent mark, if applicable, a brief description of the appearance of the diacritical and a short narrative on its usage.

acute accent; a straight diagonal line extending from upper right to lower left. The acute accent is used in most languages to lengthen a vowel; in some, such as Oscan, to denote an open vowel. The acute is also often used to indicate the stressed syllable; in some transcriptions it indicates a palatalized consonant.

diaeresis or umlaut; two dots surmounting the letter. In Romance languages and English, the diaeresis is used to indicate that consecutive vowels do not form a dipthong (see below); in modern German and Scandinavian languages, it denotes palatalization of vowels.

circumflex; a chevron or inverted 'v' shape, with the apex at the top. Used most often in modern languages to indicate lengthening of a vowel.

tilde; a curving or waving line above the letter. The tilde is a form of circumflex. The tilde is used most often in Spanish to form a palatalized n as in the word 'ano', pronounced 'anyo'. It is also used occasionally to indicate nasalized vowels.

macron; a straight line above the letter. The macron is used almost universally to lengthen a vowel.

breve; a concave semicircle or 'u' shape surmounting the letter. Originally used in Greek, the breve indicates a short vowel.

a small circle or 'o' above the letter. Frequently used in Scandinavian languages to indicate a broad 'o'.

e dipthong or ligature; transcribed as two letters in contact with each other. The dipthong is a combination of vowels that are pronounced together.

cedilla; a curved line surmounted by a vertical line, placed at the bottom of the letter. The cedilla is used in Spanish and French to denote a dental, or soft, 'c'. In the new Turkish transcription, 'c' cedilla has the value of English 'ch'. In Semitic languages, the cedilla under a consonant indicates that it is emphatic.

check or inverted circumflex; a 'v' shape above the letter. This accent is used widely in Slavic languages to indicate a palatal articulation, like the consonant sounds in the English words chapter and shoe and the 'zh' sound in pleasure.

a single dot above the letter. This diacritical denotes various things; in Lithuanian, it indicates a close long vowel. In Sanskrit, when used with 'n', it is a velar sound, as in the English 'sink'; in Irish orthography, it indicates a fricative consonant (see below).

accent grave; a diagonal line (above the letter) extending from upper left to lower right. The grave accent is used in French, Spanish and Italian to denote open vowels.

fricative; a horizontal line through a consonant. A fricative consonant is characterized by a frictional rustling of the breath as it is emitted."

END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MINIMUM_LATITUDE
DATA_TYPE = REAL
START_BYTE = 180
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The minimum_latitude element specifies the southernmost latitude of a spatial area, such as a map, mosaic, bin, feature, or region."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MAXIMUM_LATITUDE
DATA_TYPE = REAL
START_BYTE = 188
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The maximum_latitude element specifies the northernmost latitude of a spatial area, such as a map, mosaic, bin, feature, or region."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = CENTER_LATITUDE
DATA_TYPE = REAL
START_BYTE = 196
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The center latitude of the feature."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MINIMUM_LONGITUDE
DATA_TYPE = REAL

START_BYTE = 204
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The minimum_longitude element specifies the easternmost latitude of a spatial area, such as a map, mosaic, bin, feature, or region. "
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MAXIMUM_LONGITUDE
DATA_TYPE = REAL
START_BYTE = 212
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The maximum_longitude element specifies the westernmost longitude of a spatial area, such as a map, mosaic, bin, feature, or region. "
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = CENTER_LONGITUDE
DATA_TYPE = REAL
START_BYTE = 220
BYTES = 7
FORMAT = "F7.2"
UNIT = DEGREE
DESCRIPTION = "The center longitude of the feature."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = LABEL_POSITION_ID
DATA_TYPE = CHARACTER
START_BYTE = 229
BYTES = 2
FORMAT = "A2"
UNIT = "N/A"
DESCRIPTION = "The suggested plotting position of the feature name (UL=Upper left, UC=Upper center, UR=Upper right, CL=Center left, CR=Center right, LL=Lower left, LC=Lower center, LR=Lower right). This field is used to instruct the plotter where to place the typographical label with respect to the center of the feature. This code is used to avoid crowding of names in areas where there is a high density of named features."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = FEATURE_LENGTH
DATA_TYPE = REAL
START_BYTE = 233
BYTES = 8
FORMAT = "F8.2"
UNIT = KILOMETER
DESCRIPTION = "The longer or longest dimension of an object. For the Gazetteer usage, this field refers to the length of the named feature."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = PRIMARY_PARENTAGE_ID
DATA_TYPE = CHARACTER
START_BYTE = 243
BYTES = 2
FORMAT = "A2"

UNIT = "N/A"
DESCRIPTION = "This field contains the primary origin of the feature name (i.e. where the name originated). It contains a code for the continent or country origin of the name. Please see Appendix 5 of the gazetteer documentation (GAZETTER.TXT) for a definition of the codes used to define the continent or country."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SECONDARY_PARENTAGE_ID
DATA_TYPE = CHARACTER
START_BYTE = 248
BYTES = 2
FORMAT = "A2"
UNIT = "N/A"
DESCRIPTION = "This field contains the secondary origin of the feature name. It contains a code for a country, state, territory, or ethnic group. Please see Appendix 5 of the gazetteer documentation (GAZETTER.TXT) for a defintion of the codes in this field."

END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MAP_SERIAL_ID
DATA_TYPE = CHARACTER
START_BYTE = 253
BYTES = 6
FORMAT = "A6"
UNIT = "N/A"
DESCRIPTION = "The identification of the map that contains the named feature. This field represents the map serial number of the map publication used for ordering maps from the U.S. Geological Survey. The map identified in this field best portrays the named feature."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = FEATURE_STATUS_TYPE
DATA_TYPE = CHARACTER
START_BYTE = 262
BYTES = 12
FORMAT = "A12"
UNIT = "N/A"
DESCRIPTION = "The IAU approval status of the named feature. Permitted values are 'PROPOSED', 'PROVISIONAL', 'IAU-APPROVED', and 'DROPPED'. Dropped names have been disallowed by the IAU. However, these features have been included in the gazetteer for historical purposes. Some named features that are disallowed by the IAU may commonly be used on some maps."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = APPROVAL_DATE
DATA_TYPE = INTEGER
START_BYTE = 276
BYTES = 4
FORMAT = "I4"
UNIT = "N/A"
DESCRIPTION = "Date at which an object has been approved by the officially sanctioned organization. This field contains the year the IAU approved the feature name."
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = FEATURE_TYPE
DATA_TYPE = CHARACTER
START_BYTE = 282

BYTES = 20
FORMAT = "A20"
UNIT = "N/A"
DESCRIPTION = "The feature type identifies the type of a particular feature, according to IAU standards. Examples are 'CRATER', 'TESSERA', 'TERRA', etc. See Appendix 7 of the gazetteer documentation (GAZETTER.TXT).
DESCRIPTOR TERMS (FEATURE TYPES)

FEATURE	DESCRIPTION
ALBEDO FEATURE	Albedo feature
CATENA	Chain of craters
CAVUS	Hollows, irregular depressions
CHAOS	Distinctive area of broken terrain
CHASMA	Canyon
COLLES	Small hill or knob
CORONA	Ovoid-shaped feature
CRATER	Crater
DORSUM	Ridge
ERUPTIVE CENTER	Eruptive center
FACULA	Bright spot
FLEXUS	Cuspate linear feature
FLUCTUS	Flow terrain
FOSSA	Long, narrow, shallow depression
LABES	Landslide
LABYRINTHUS	Intersecting valley complex
LACUS	Lake
LARGE RINGED FEATURE	Large ringed feature
LINEA	Elongate marking
MACULA	Dark spot
MARE	Sea
MENSA	Mesa, flat-topped elevation
MONS	Mountain
OCEANUS	Ocean
PALUS	Swamp
PATERA	Shallow crater; scalloped, complex edge
PLANITIA	Low plain
PLANUM	Plateau or high plain
PROMONTORIUM	Cape
REGIO	Region
RIMA	Fissure
RUPES	Scarp

SCOPULUS	Lobate or irregular scarp
SINUS	Bay
SULCUS	Subparallel furrows and ridges
TERRA	Extensive land mass
TESSERA	Tile; polygonal ground
THOLUS	Small domical mountain or hill
UNDAE	Dunes
VALLIS	Sinuous valley
VASTITAS	Widespread lowlands
VARIABLE FEATURE	Variable feature "

```
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
NAME                            = REFERENCE_NUMBER
DATA_TYPE                       = INTEGER
START_BYTE                      = 304
BYTES                           = 4
FORMAT                          = "I4"
UNIT                            = "N/A"
DESCRIPTION                     = "Literature reference from which the spelling and description of the feature
name was derived.  See Appendix 6 of the gazetteer documentation  (GAZETTER.TXT)."
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
NAME                            = MAP_CHART_ID
DATA_TYPE                       = CHARACTER
START_BYTE                      = 310
BYTES                           = 6
FORMAT                          = "A6"
UNIT                            = "N/A"
DESCRIPTION                     = "This field contains the abbreviation of the map designator or chart
identification (example MC-19, MC-18, etc.)."
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
NAME                            = FEATURE_DESCRIPTION
DATA_TYPE                       = CHARACTER
START_BYTE                      = 319
BYTES                           = 159
FORMAT                          = "A159"
UNIT                            = "N/A"
DESCRIPTION                     = "Short description of the feature name."
END_OBJECT                      = COLUMN
END_OBJECT                      = GAZETTEER_TABLE

END
```

A.16 HEADER

The HEADER object is used to identify and define the attributes of commonly used header data structures such as VICAR or FITS. These structures are usually system or software specific and are described in detail in a referenced description text file. The use of bytes within the header object refers to the number of bytes for the entire header, not a single record.

Required Keywords

1. BYTES
2. HEADER_TYPE

Optional Keywords

1. DESCRIPTION
2. INTERCHANGE_FORMAT
3. RECORDS

Required Objects

None

Optional Objects

None

Example

The following example shows the detached label file "TIMTC02A.LBL". The label describes the data product file "TIMTC02A.IMG" which contains a HEADER object followed by an IMAGE object.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
/* PDS label for a TIMS image */
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 638
FILE_RECORDS                    = 39277
/* Pointers to objects */
^IMAGE_HEADER                   = ("TIMTC02A.IMG",1)
^IMAGE                          = ("TIMTC02A.IMG",2)
/* Image description */
DATA_SET_ID                     = "C130-E-TIMS-2-EDR-IMAGE-V1.0"
PRODUCT_ID                      = "TIMTC02A"
INSTRUMENT_HOST_NAME            = "NASA C-130 AIRCRAFT"
INSTRUMENT_NAME                 = "THERMAL INFRARED MULTISPECTRAL SCANNER"
TARGET_NAME                     = EARTH
FEATURE_NAME                    = "TRAIL CANYON FAN"
START_TIME                      = 1989-09-29T21:47:35Z
```

```
STOP_TIME                       = 1989-09-29T21:47:35Z
CENTER_LATITUDE                 = 36.38
CENTER_LONGITUDE                = 116.96
INCIDENCE_ANGLE                 = 0.0
EMISSION_ANGLE                  = 0.0
/* Description of objects */
OBJECT                          = IMAGE_HEADER
BYTES                           = 638
RECORDS                         = 1
HEADER_TYPE                     = VICAR2
INTERCHANGE_FORMAT              = BINARY
^DESCRIPTION                    = "VICAR2.TXT"
END_OBJECT                      = IMAGE_HEADER

OBJECT                          = IMAGE
LINES                           = 6546
LINE_SAMPLES                    = 638
SAMPLE_TYPE                     = UNSIGNED_INTEGER
SAMPLE_BITS                     = 8
SAMPLE_BIT_MASK                 = 2#11111111#
BANDS                           = 6
BAND_STORAGE_TYPE               = LINE_INTERLEAVED
END_OBJECT                      = IMAGE
END
```

A.17 HISTOGRAM

The HISTOGRAM object is a sequence of numeric values that provides the number of occurrences of a data value or a range of data values in a data object. The number of items in a histogram will normally be equal to the number of distinct values allowed in a field of the data object. For example, an 8 bit integer field can have a maximum of 256 values, and would result in a 256 item histogram. Histograms may be used to bin data, in which case an offset and scaling factor indicate the dynamic range of the data represented.

The following equation allows the calculation of the range of each 'bin' in the histogram.

'bin lower boundary' = 'bin element' * scale_factor + offset

Required Keywords

1. ITEMS
2. DATA_TYPE
3. ITEM_BYTES

Optional Keywords

1. BYTES
2. INTERCHANGE_FORMAT
3. OFFSET
4. SCALING_FACTOR

Required Objects

None

Optional Objects

None

Example

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3
/*      FILE FORMAT AND LENGTH */

RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 956
FILE_RECORDS                            = 965
LABEL_RECORDS                           = 3

/*      POINTERS TO START RECORDS OF OBJECTS IN FILE */

^IMAGE_HISTOGRAM                        = 4
```

```
^IMAGE                                  = 6

/*      IMAGE DESCRIPTION */

DATA_SET_ID                             = "VO1/VO2-M-VIS-5-DIM-V1.0"
PRODUCT_ID                              = "MG15N022-GRN-666A"
SPACECRAFT_NAME                         = VIKING_ORBITER_1
TARGET_NAME                             = MARS
START_TIME                              = 1978-01-14T02:00:00
STOP_TIME                               = 1978-01-14T02:00:00
SPACECRAFT_CLOCK_START_TIME             = UNK
SPACECRAFT_CLOCK_STOP_TIME              = UNK
PRODUCT_CREATION_TIME                   = 1995-01-01T00:00:00
ORBIT_NUMBER                            = 666
FILTER_NAME                             = GREEN
IMAGE_ID                                = "MG15N022-GRN-666A"
INSTRUMENT_NAME                         = {VISUAL_IMAGING_SUBSYSTEM_CAMERA_A,
VISUAL_IMAGING_SUBSYSTEM_CAMERA_B}
NOTE                                    = "MARS MULTI-SPECTRAL MDIM SERIES"
/* SUN RAYS EMISSION, INCIDENCE, AND PHASE ANGLES OF IMAGE CENTER*/
SOURCE_PRODUCT_ID                        =" 666A36"
EMISSION_ANGLE                          = 21.794
INCIDENCE_ANGLE                         = 66.443
PHASE_ANGLE                             = 46.111

/*      DESCRIPTION OF OBJECTS CONTAINED IN FILE */

OBJECT                                  = IMAGE_HISTOGRAM
ITEMS                                   = 256
DATA_TYPE                               = VAX_INTEGER
ITEM_BYTES                              = 4
END_OBJECT                              = IMAGE_HISTOGRAM

OBJECT                                  = IMAGE
LINES                                   = 960
LINE_SAMPLES                            = 956
SAMPLE_TYPE                             = UNSIGNED_INTEGER
SAMPLE_BITS                             = 8
SAMPLE_BIT_MASK                         = 2#11111111#
CHECKSUM                                = 65718982
/*  I/F = SCALING_FACTOR*DN + OFFSET, CONVERT TO INTENSITY/FLUX */
SCALING_FACTOR                          = 0.001000
OFFSET                                  = 0.0
/* OPTIMUM COLOR STRETCH FOR DISPLAY OF COLOR IMAGES */
STRETCHED_FLAG                          = FALSE
STRETCH_MINIMUM                         = ( 53,  0)
STRETCH_MAXIMUM                         = (133,255)
END_OBJECT                              = IMAGE

END
```

A.18 HISTORY

A HISTORY object is a dynamic description of the history of one or more associated data objects in a file. It supplements the essentially static description contained in the PDS label.

The HISTORY object contains text in a format similar to that of the ODL statements used in the label. It identifies previous computer manipulation of the principal data object(s) in the file. It includes an identification of the source data, processes performed, processing parameters, as well as dates and times of processing. It is intended that the history be available for display, be dynamically extended by any process operating on the data, and automatically propagated to the resulting data file. Eventually, it might be extracted for loading in detailed level catalogs of data set contents.

The HISTORY object is structured as a series of History Entries, one for each process which has operated on the data. Each entry contains a standard set of ODL element assignment statements, delimited by GROUP = program_name and END_GROUP = program_name statements. A subgroup in each entry, delimited by GROUP = PARAMETERS and END_GROUP = PARAMETERS, contains statements specifying the values of all parameters of the program.

HISTORY ENTRY ELEMENTS

Attribute	Description
VERSION_DATE	Program version date, ISO standard format.
DATE_TIME	Run date and time, ISO standard format.
NODE_NAME	Network name of computer.
USER_NAME	Username.
SOFTWARE_DESC	Program-generated (brief) description.
USER_NOTE	User-supplied (brief) description.

Unlike the above elements, the names of the parameters defined in the PARAMETERS subgroup are uncontrolled, and must only conform to the program.

The last entry in a HISTORY object is followed by an END statement. The HISTORY object, by convention, follows the PDS label of the file, beginning on a record boundary, and is located by a pointer statement in the label. There are no required elements for the PDS label description of the object; it is represented in the label only by the pointer statement, and OBJECT = HISTORY and END_OBJECT = HISTORY statements.

The HISTORY capability has been implemented as part of the Integrated Software for Imaging Spectrometers (ISIS) system (see QUBE object definition). ISIS Qube applications add their own entries to the Qube file's cumulative History object. ISIS programs run under NASA's TAE (Transportable Applications Executive) system, and are able to automatically insert all parameters of their TAE procedure into the history entry created by the program. Consult the ISIS System Design document for details and limitations imposed by that system. (See the QUBE object description for further references.)

Required Keywords

None

Optional Keywords

None

Required Objects

None

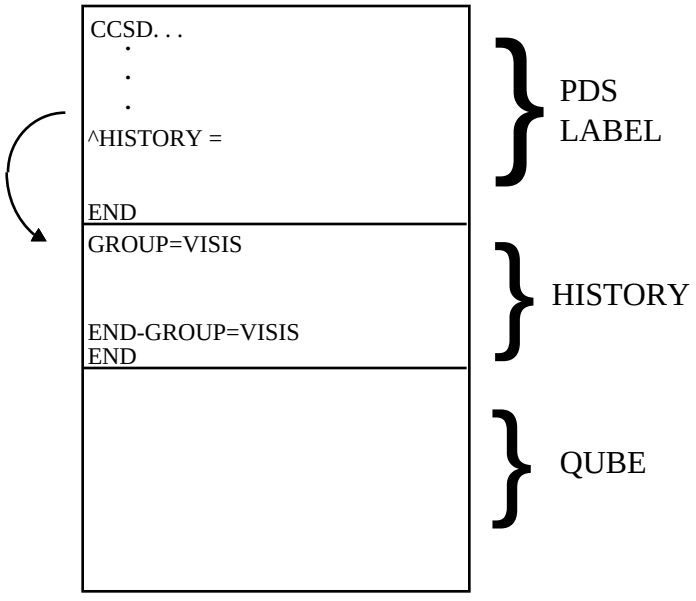
Optional Objects

None

Example

The following single-entry HISTORY object is from a Vicar-generated PDS-labeled qube file. (See the Qube object example.) There's only one entry because the qube (or rather its label) was generated by a single program, VISIS. A qube generated by multiple ISIS programs would have multiple history entries, represented by multiple GROUPs in the HISTORY object.

This diagram illustrates the placement of the example HISTORY object within a Qube data product with an attached PDS label.




```
GROUP                          = VISIS

  VERSION_DATE                 = 1990-11-08
```

```
  DATE_TIME                          = 1991-07-25T10:12:52
  SOFTWARE_DESC                      = "ISIS cube file with PDS label has been generated as systematic product by
MIPL using the following programs:
      NIMSMERGE to create EDR's;
      NIMSCMM to create the merged mosaic & geometry cube;
      HIST2D to create a two-dimensional histogram;
      SPECPLOT to create the spectral plots;
      TRAN, F2, and INSERT3D to create the SII cube;
      VISIS to create the ISIS cube."

  USER_NOTE                          = "VPDIN1/ Footprint, Limbfit, Height=50"

  GROUP                              = PARAMETERS
   EDR_FILE_NAME                     = " "          /*EDR accessed through MIPL Catalog*/
    IMAGE_ID                         = NULL
    SPICE_FILE_NAME                  = " "
    SPIKE_FILE_NAME                  = "mipl:[mipl.gll]boom_obscuration.nim"
    DARK_VALUE_FILE_NAME             = " "
    CALIBRATION_FILE_NAME            = "ndat:nimsgs2.cal"
    MERGED_MOSAIC_FILE_NAME          = "ndat:vpdin1_dn_fp_lf_h50.CUB"
    DARK_INTERPOLATION_TYPE          = NOUPDAT
    PHOTOMETRIC_CORRECTION_TYPE      = NONE
    CUBE_NIMSEL_TYPE                 = NOCAL
    BINNING_TYPE                     = FOOTPRNT
    FILL_BOX_SIZE                    = 0
    FILL_MIN_VALID_PIXELS            = 0
    SUMMARY_IMAGE_RED_ID             = 0
    SUMMARY_IMAGE_GREEN_ID           = 0
    SUMMARY_IMAGE_BLUE_ID            = 0
    ADAPT_STRETCH_SAT_FRAC           = 0.000000
    ADAPT_STRETCH_SAMP_FRAC          = 0.000000
    RED_STRETCH_RANGE                = (    0,    0)
    GREEN_STRETCH_RANGE              = (    0,    0)
    BLUE_STRETCH_RANGE               = (    0,    0)
  END_GROUP                          = PARAMETERS
 END_GROUP                           = VISIS
 END
```

A.19 IMAGE

An IMAGE object is an array of sample values. Image objects are normally processed with special display tools to produce a visual representation of the sample values. This is done by assigning brightness levels or display colors to the various sample values. Images are composed of LINES and SAMPLES. They may contain multiple bands, in one of several storage orders.

Simple IMAGE objects are defined as having LINES as the number of horizontal lines, with each line having LINE_SAMPLES as the number of sample values defined. The default sample values are 8-bit unsigned binary integer. The sample size can be over- ridden using the SAMPLE_BITS keyword (e.g. SAMPLE_BITS = 32). The SAMPLE_TYPE keyword can be used to override the default SAMPLE_TYPE (e.g. SAMPLE_TYPE = VAX_REAL).

Each line of an IMAGE object may also be organized with a set of PREFIX or SUFFIX bytes, which provide engineering parameters related to each line. The PREFIX or SUFFIX area is treated as a TABLE object which has been concatenated with the IMAGE object. Each physical record in the file contains a row of the PREFIX or SUFFIX table and a line of the IMAGE. While this is a commonly used format for IMAGE storage, it can cause difficulties if used with general purpose display and processing software. In particular, most programs will consider the PREFIX and SUFFIX as part of the image, meaning that statistics generated for the image (mean, standard deviation, etc.) will be in error. It is recommended that PREFIX or SUFFIX information be stored as a separate TABLE data object in separate records within the file and not concatenated with the image data. (See Figure A.1.)

Most images are composed of LINES containing a horizontal array of SAMPLES. However some imaging sensors may scan in a vertical direction, creating an array of vertical lines, as in the case of the Viking Lander camera system.

More complex IMAGE formats include multi-band images, where SAMPLES or LINES of the same scene from several spectral bands are combined in one object, by sample (SAMPLE_INTERLEAVED), or by line (LINE_INTERLEAVED). Another IMAGE format is TILED, where a large IMAGE is divided into smaller pieces (TILES) to provide efficient access.

Figure A.2 illustrates the BANDS, BAND_NAME, and BAND_STORAGE_TYPE keywords that can be used to describe multi-band images.

Note: Additional engineering values may be prepended or appended to each LINE of an image, and are stored as concatenated TABLE objects, which must be named LINE_PREFIX and LINE_SUFFIX. IMAGE objects may be associated with other objects, including HISTOGRAMs, PALETTEs, HISTORY and TABLEs which contain statistics, display parameters, engineering values or other ancillary data.

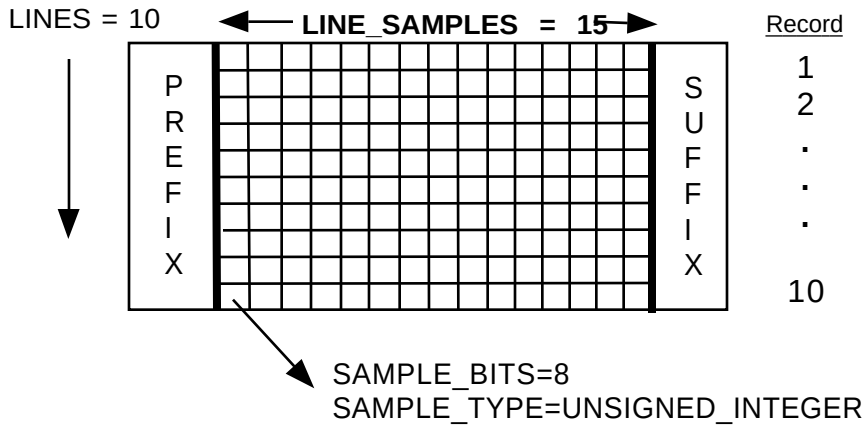


Figure A.1: Prefix and Suffix Bytes attached to an Image

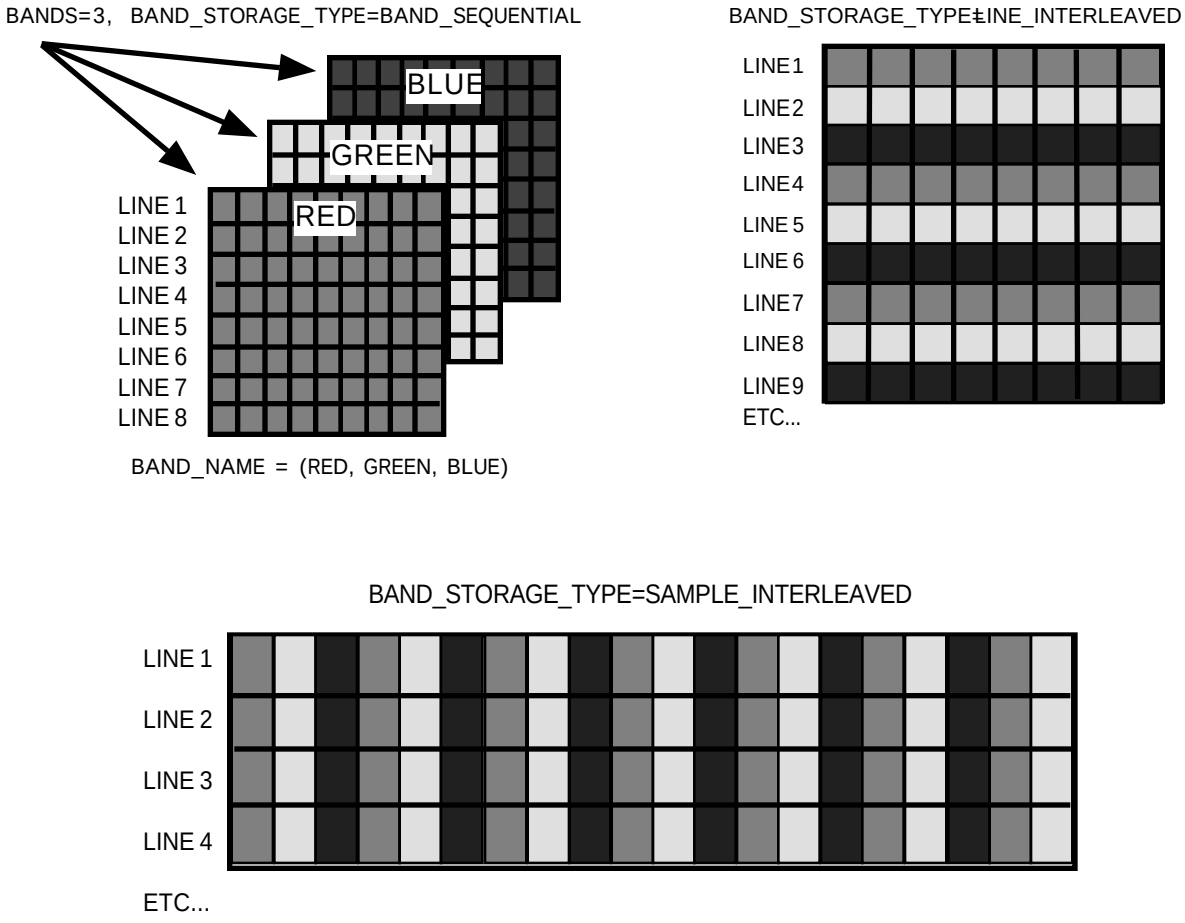


Figure A.2: Keywords for a Multi-Band Image

Required Keywords

1. LINES
2. LINE_SAMPLES
3. SAMPLE_TYPE
4. SAMPLE_BITS

Optional Keywords

1. BAND_SEQUENCE
2. BAND_STORAGE_TYPE
3. BANDS
4. CHECKSUM
5. DERIVED_MAXIMUM
6. DERIVED_MINIMUM
7. DESCRIPTION
8. ENCODING_TYPE
9. FIRST_LINE
10. FIRST_LINE_SAMPLE
11. INVALID_CONSTANT
12. LINE_PREFIX_BYTES
13. LINE_SUFFIX_BYTES
14. MISSING _CONSTANT
15. OFFSET
16. SAMPLE_BIT_MASK
17. SAMPLING_FACTOR
18. SCALING_FACTOR
19. SOURCE_FILE_NAME
20. SOURCE_LINES
21. SOURCE_LINE_SAMPLES
22. SOURCE_SAMPLE_BITS
23. STRETCHED_FLAG
24. STRETCH_MINIMUM
25. STRETCH_MAXIMUM

Required Objects

None

Optional Objects

None

Example

This is an example of an attached IMAGE label for a color digital mosaic image from the Mars Digital Image Map CD-ROMs. It includes a CHECKSUM to support automated volume production and validation, a SCALING_FACTOR to indicate the relationship between sample values and geophysical parameters and stretch keywords to indicate optimal values for image display.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3

/*  FILE FORMAT AND LENGTH */

RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 956
FILE_RECORDS                    = 965
LABEL_RECORDS                   = 3

/*  POINTERS TO START RECORDS OF OBJECTS IN FILE */

^IMAGE_HISTOGRAM                = 4
^IMAGE                          = 6

/*  IMAGE DESCRIPTION */

DATA_SET_ID                     = "VO1/VO2-M-VIS-5-DIM-V1.0"
PRODUCT_ID                      = "MG15N022-GRN-666A"
SPACECRAFT_NAME                 = VIKING_ORBITER_1
TARGET_NAME                     = MARS
IMAGE_TIME                      = 1978-01-14T02:00:00
START_TIME                      = UNK
STOP_TIME                       = UNK
SPACECRAFT_CLOCK_START_COUNT    = UNK
SPACECRAFT_CLOCK_STOP_COUNT     = UNK
PRODUCT_CREATION_TIME           = 1995-01-01T00:00:00
ORBIT_NUMBER                    = 666
FILTER_NAME                     = GREEN
IMAGE_ID                        = "MG15N022-GRN-666A"
INSTRUMENT_NAME                 = {VISUAL_IMAGING_SUBSYSTEM_CAMERA_A,
                                  VISUAL_IMAGING_SUBSYSTEM_CAMERA_B}
NOTE                            = "MARS MULTI-SPECTRAL MDIM SERIES"
SOURCE_PRODUCT_ID               = "666A36"
EMISSION_ANGLE                  = 21.794
INCIDENCE_ANGLE                 = 66.443
PHASE_ANGLE                     = 46.111

/*  DESCRIPTION OF OBJECTS CONTAINED IN FILE */

OBJECT                          = IMAGE_HISTOGRAM
ITEMS                           = 256
DATA_TYPE                       = VAX_INTEGER
ITEM_BYTES                      = 4
END_OBJECT                      = IMAGE_HISTOGRAM
```

```
OBJECT                          = IMAGE
LINES                           = 960
LINE_SAMPLES                    = 956
SAMPLE_TYPE                     = UNSIGNED_INTEGER
SAMPLE_BITS                     = 8
SAMPLE_BIT_MASK                 = 2#11111111#
CHECKSUM                        = 65718982
SCALING_FACTOR                  = 0.001000     /* I/F = scaling factor * DN + offset, */
                                  /* convert to intensity/flux.          */
OFFSET                          = 0.0
STRETCHED_FLAG                  = FALSE        /* Optimum color stretch for display   */
STRETCH_MINIMUM                 = ( 53,  0)    /* of color images.                    */
STRETCH_MAXIMUM                 = (133,255)
END_OBJECT                      = IMAGE

END
```

A.20 IMAGE MAP PROJECTION

The IMAGE_MAP_PROJECTION object is one of two distinct objects that define the map projection used in creating the digital images in a PDS data set.The name of the other associated object that completes the definition is called DATA_SET_MAP_PROJECTION.(see Appendix B)

The map projection information resides in these two objects, essentially to reduce data redundancy and at the same time allow the inclusion of elements needed to process the data at the image level. Basically, static information that is applicable to the complete data set reside in the DATA_SET_MAP_PROJECTION object, while dynamic information that is applicable to the individual images reside in the IMAGE_MAP_PROJECTION object.

The line_first_pixel, line_last_pixel, sample_first_pixel, and sample_last_pixel keywords are used to indicate which way is up in an image. Sometimes an image can be shifted or flipped prior to it being physically recorded. These keywords are used in calculating the mapping of pixels between the original image and the stored image.

The following equations give the byte offsets needed to determine the mapping of a pixel (X,Y) from the original image to a pixel in the stored image:

The sample offset from the first pixel is:

$$\frac{\text{sample\_bits} * (Y - \text{sample\_first\_pixel}) * \text{line\_samples}}{8 * (\text{sample\_last\_pixel} - \text{sample\_first\_pixel} + 1)}$$

The line offset from the first image line is:

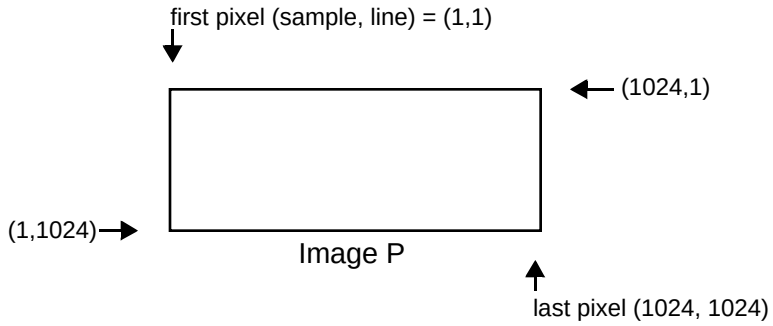
$$\frac{(X - \text{line\_first\_pixel}) * \text{lines}}{(\text{line\_last\_pixel} - \text{line\_first\_pixel} + 1)}$$

Additionally, in any image, ABS (sample_last_pixel - sample_first_pixel + 1) is always equal to line_samples, and ABS (line_last_pixel - line_first_pixel + 1) is always equal to lines.

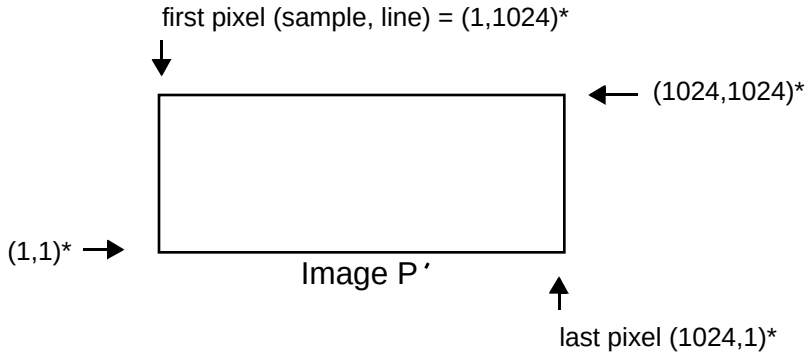
Example

Take a 1K by 1K 8-bit image which is rotated about the x-axis 180 degrees prior to being physically recorded.

Original Image: Positive direction is to the right and down



Stored Image: Positive direction is to the right and up



These pixel location values (*) are the positions from the original image. For example, the first pixel in the stored image (normally referred to as (1,1)) came from the position (1,1024) in the original image. These original values are used for the following IMAGE_MAP_PROJECTION keywords in the PDS label for the stored image:

sample_first_pixel = 1
sample_last_pixel = 1024
line_first_pixel = 1024
line_last_pixel = 1

Now, given a pixel on the original image, P(X,Y) = (2,2) determine its location (P') in the stored image.

sample offset = (8 * (2 - 1) * 1024) / (8 * (1024 - 1 + 1)) = 1
line offset = ((2 - 1024) * 1024) / (1 - 1024 + 1) = (-1022)

Therefore, P' is located at (2, 1023) which is 1 byte from the first sample, and 1022 bytes (in the negative direction) from the first line in the stored image. See diagram above.

Required Keywords

1. MAP_PROJECTION_TYPE
2. A_AXIS_RADIUS
3. B_AXIS_RADIUS
4. C_AXIS_RADIUS
5. FIRST_STANDARD_PARALLEL
6. SECOND_STANDARD_PARALLEL
7. POSITIVE_LONGITUDE_DIRECTION
8. CENTER_LATITUDE
9. CENTER_LONGITUDE
10. REFERENCE_LATITUDE
11. REFERENCE_LONGITUDE
12. LINE_FIRST_PIXEL
13. LINE_LAST_PIXEL
14. SAMPLE_FIRST_PIXEL
15. SAMPLE_LAST_PIXEL
16. MAP_PROJECTION_ROTATION
17. MAP_RESOLUTION
18. MAP_SCALE
19. MAXIMUM_LATITUDE
20. MINIMUM_LATITUDE
21. EASTERNMOST_LONGITUDE
22. WESTERNMOST_LONGITUDE
23. LINE_PROJECTION_OFFSET
24. SAMPLE_PROJECTION_OFFSET
25. COORDINATE_SYSTEM_TYPE
26. COORDINATE_SYSTEM_NAME

Optional Keywords

1. DATA_SET_ID
2. IMAGE_ID
3. HORIZONTAL_FRAMELET_OFFSET
4. VERTICAL_FRAMELET_OFFSET

Required Objects

1. DATA_SET_MAP_PROJECTION

Optional Objects

None

Example

```
PDS_VERSION_ID                  = PDS3

/* File characteristics */
RECORD_TYPE                     = STREAM

/* Identification data elements */
DATA_SET_ID                      = "MGN-V-RDRS-5-GVDR-V1.0"
DATA_SET_NAME                   = "MAGELLAN VENUS RADAR SYSTEM GLOBAL DATA RECORD
V1.0"
PRODUCT_ID                      = "IMP-NORTH.100"

MISSION_NAME                    = "MAGELLAN"
SPACECRAFT_NAME                 = "MAGELLAN"
INSTRUMENT_NAME                 = "RADAR SYSTEM"
TARGET_NAME                     = "VENUS"

ORBIT_START_NUMBER              = 376
ORBIT_STOP_NUMBER               = 4367
START_TIME                      = "N/A"
STOP_TIME                       = "N/A"
SPACECRAFT_CLOCK_START_COUNT    = "N/A"
SPACECRAFT_CLOCK_STOP_COUNT     = "N/A"

PRODUCT_CREATION_TIME            = 1994-05-07T22:09:27.000
PRODUCT_RELEASE_DATE            = 1994-05-13
PRODUCT_SEQUENCE_NUMBER         = 00000
PRODUCT_VERSION_TYPE            = "PRELIMINARY"

SOURCE_DATA_SET_ID               = {"MGN-V-RDRS-5-SCVDR-V1.0",
"MGN-V-RDRS-CDR-ALT/RAD-V1.0"}
SOURCE_PRODUCT_ID               = {"SCVDR.00376-00399.1","SCVDR.00400-00499.1",
"SCVDR.01100-01199.1","SCVDR.01200-01299.1","SCVDR.01300-01399.1",
"SCVDR.01400-01499.1","SCVDR.01500-01599.1","SCVDR.01600-01699.1",
"SCVDR.01700-01799.1","SCVDR.01800-01899.1","SCVDR.01900-01999.1",
"ARCDRCD.001;2","ARCDRCD.002;1","ARCDRCD.003;1","ARCDRCD.004;1",
"ARCDRCD.005;1","ARCDRCD.006;1","ARCDRCD.007;1","ARCDRCD.008;1",
"ARCDRCD.017;1","ARCDRCD.018;1","ARCDRCD.019;1"}

SOFTWARE_FLAG                   = "Y"

PRODUCER_FULL_NAME              = "MICHAEL J. MAURER"
PRODUCER_INSTITUTION_NAME       = "STANFORD CENTER FOR RADAR ASTRONOMY"
PRODUCER_ID                     = "SCRA"
DESCRIPTION                      = "This file contains a single
  IMAGE_MAP_PROJECTION data object with an attached PDS label."

/* Data object definitions */
OBJECT                          = IMAGE_MAP_PROJECTION
  ^DATA_SET_MAP_PROJECTION      = "DSMAP.CAT"
  COORDINATE_SYSTEM_TYPE        = "BODY-FIXED ROTATING"
  COORDINATE_SYSTEM_NAME        = "PLANETOCENTRIC"
```

```
  MAP_PROJECTION_TYPE                = "STEREOGRAPHIC"
  A_AXIS_RADIUS                      = 6051.0 <KM>
  B_AXIS_RADIUS                      = 6051.0 <KM>
  C_AXIS_RADIUS                      = 6051.0 <KM>
  FIRST_STANDARD_PARALLEL            = "N/A"
  SECOND_STANDARD_PARALLEL           = "N/A"
  POSITIVE_LONGITUDE_DIRECTION       = "EAST"
  CENTER_LATITUDE                    = 90
  CENTER_LONGITUDE                   = 0
  REFERENCE_LATITUDE                 = "N/A"
  REFERENCE_LONGITUDE                = "N/A"
  LINE_FIRST_PIXEL                   = 1
  LINE_LAST_PIXEL                    = 357
  SAMPLE_FIRST_PIXEL                 = 1
  SAMPLE_LAST_PIXEL                  = 357
  MAP_PROJECTION_ROTATION            = 0
  MAP_RESOLUTION                     = 5.79478 <PIXEL/DEGREE>
  MAP_SCALE                          = 18.225 <KM/PIXEL>
  MAXIMUM_LATITUDE                   = 90.00
  MINIMUM_LATITUDE                   = 60.00
  EASTERNMOST_LONGITUDE              = 360.00
  WESTERNMOST_LONGITUDE              = 0.00
  LINE_PROJECTION_OFFSET             = 178
  SAMPLE_PROJECTION_OFFSET           = 178
END_OBJECT                           = IMAGE_MAP_PROJECTION
END
```

A.21 INDEX_TABLE

The INDEX_TABLE object is a specific type of a TABLE object that provides information about the data stored on an archive volume. The INDEX_TABLE contains one row for each data file (or data product label file, in the case where detached labels are used) on the volume. The table is formatted so that it may be read directly by many data management systems on various host computers. All fields (columns) are separated by commas, and character fields are enclosed by double quotation marks. Each record ends in a carriage return/line feed sequence. This allows the table to be treated as a fixed length record file on hosts that support this file type and as a normal text file on other hosts.

There are two categories of columns for an Index table, identification and search. PDS data element names should be used as column names wherever appropriate.

The required columns are used for identification. The optional columns are data dependent and are used for search. For example, the following may be useful for searching:

Location (e.g. LATITUDE, LONGITUDE, ORBIT_NUMBER)
Time (e.g. START_TIME, SPACECRAFT_CLOCK_START_COUNT)
Feature (e.g. FEATURE_TYPE)
Observational characteristics (e.g. INCIDENCE_ANGLE)
Instrument characteristics (e.g. FILTER_NAME)

For archive volumes created before version 3.2 of the PDS standards, if the keyword INDEX_TYPE is not present, the value is defaulted to SINGLE, unless the Index's filename is given as CUMINDEX.TAB or axxCMIDX.TAB (with axx representing up to three alphanumeric characters).

If the keyword INDEXED_FILE_NAME is not present for a SINGLE index, the value is defaulted to "*.*" if attached labels are used, or "*.LBL" if detached labels are used. This indicates that the index encompasses all data product files on the volume.

If the INDEXED_FILE_NAME keyword is not present for a cumulative index, the default value is "*.TAB" for files in the INDEX subdirectory.

Note: See section 17.2 for information about the use of N/A, UNK and NULL in an INDEX table.

Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES
5. INDEX_TYPE

Optional Keywords

1. NAME
2. DESCRIPTION
3. INDEXED_FILE_NAME
4. UNKNOWN_CONSTANT
5. NOT_APPLICABLE_CONSTANT

Required Objects

1. COLUMN

Optional Objects

None

Required COLUMN Objects (NAME=)

FILE_SPECIFICATION_NAME or PATH_NAME and FILE_NAME
PRODUCT_ID (**)
VOLUME_ID (*)
DATA_SET_ID (*)
PRODUCT_CREATION_TIME (*)
LOGICAL_VOLUME_PATH_NAME (must be used with PATH_NAME and FILE_NAME for a logical volume) (*)

(*) If the value is constant across the data in the index table, this keyword can appear in the index table's label.
If the value is not constant, then a column of the given name must be used.

(**) PRODUCT_ID is not required if it has the same value as FILE_NAME or FILE_SPECIFICATION_NAME.

Required Keywords (for Required COLUMN Objects)

NAME
DATA_TYPE
START_BYTE
BYTES
DESCRIPTION

Optional COLUMN Objects (NAME=)

MISSION_NAME
INSTRUMENT_NAME (or ID)
INSTRUMENT_HOST_NAME (or ID) (or SPACECRAFT_NAME or ID)
TARGET_NAME
PRODUCT_TYPE
MISSION_PHASE_NAME
VOLUME_SET_ID
START_TIME
STOP_TIME
SPACECRAFT_CLOCK_START_COUNT
SPACECRAFT_CLOCK_STOP_COUNT
any other search columns

Example (see additional example in A.27.1)

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3

RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 180
FILE_RECORDS                            = 220
DESCRIPTION                             = "INDEX.TAB lists all data files on this volume"
^INDEX_TABLE                            = "INDEX.TAB"

OBJECT                                  = INDEX_TABLE
  INTERCHANGE_FORMAT                    = ASCII
  ROW_BYTES                             = 180
  ROWS                                  = 220
  COLUMNS                               = 9
  INDEX_TYPE                            = SINGLE
  INDEXED_FILE_NAME                     = {"*.AMD","*.ION","*.TIM","*.TRO",
                                          "*.WEA","*.LIT","*.MIF","*.MPD",
                                           "*.ODF","*.ODR","*.ODS","*.SFO",
                                           "*.SOE","*.TDF"}

  OBJECT                                = COLUMN
    NAME                                = VOLUME_ID
    DESCRIPTION                         = "Identifies the volume containing  the named file"
    DATA_TYPE                           = CHARACTER
    START_BYTE                          = 2
    BYTES                               = 9
  END_OBJECT                            = COLUMN

  OBJECT                                = COLUMN
    NAME                                = DATA_SET_ID
    DESCRIPTION                         = "The data set identifier. Acceptable values include
                                           'MO-M-RSS-1-OIDR-V1.0'"
    DATA_TYPE                           = CHARACTER
    START_BYTE                          = 14
    BYTES                               = 25
  END_OBJECT                            = COLUMN

  OBJECT                                = COLUMN
    NAME                                 = PATH_NAME
    DESCRIPTION                         = "Path to directory containing file.
                                          Acceptable values include:
                                           'AMD ',
                                           'ION ',
                                           'TIM ',
                                           'TRO',
                                           'WEA ',
                                           'LIT  ',
                                           'MIF ',
                                           'MPD ',
                                           'ODF ',
                                          'ODR ',
                                          'ODS ',
                                           'SFO ',
                                           'SOE', and
                                           'TDF '."
```

```
    DATA_TYPE                  = CHARACTER
    START_BYTE                 = 42
    BYTES                       = 9
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = FILE_NAME
    DESCRIPTION                 = "Name of file in archive"
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 54
    BYTES                       = 12
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = PRODUCT_ID
    DESCRIPTION                 = "Original file name on MO PDB or  SOPC"
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 69
    BYTES                       = 33
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = START_TIME
    DESCRIPTION                 = "Time at which data in the file begin given in the format
                                  'YYYY-MM-DDThh:mm:ss'."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 105
    BYTES                       = 19
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = STOP_TIME
    DESCRIPTION                 = "Time at which data in the file end  given in the format
                                   'YYYY-MM-DDThh:mm:ss'."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 127
    BYTES                       = 19
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = PRODUCT_CREATION_TIME
    DESCRIPTION                 = "Date and time that file was created."
    DATA_TYPE                   = CHARACTER
    START_BYTE                  = 149
    BYTES                       = 19
  END_OBJECT                    = COLUMN

  OBJECT                        = COLUMN
    NAME                        = FILE_SIZE
    DESCRIPTION                 = "Number of bytes in file, not  including label."
    DATA_TYPE                   = "ASCII INTEGER"
    START_BYTE                  = 170
    BYTES                       = 9
  END_OBJECT                    = COLUMN

END_OBJECT                      = INDEX_TABLE
END
```

A.22 PALETTE

The PALETTE object, a sub-class of the table object, contains entries which represent color assignments for SAMPLE values contained in an IMAGE.

If the palette is stored in an external file from the data file, then the palette should be stored in ASCII format as 256 ROWS, each composed of 4 COLUMNS. The first column contains the SAMPLE value (0 to 255 for an 8-bit SAMPLE), and the remaining 3 COLUMNS contains the relative amount (a value from 0 to 255) of each primary color to be assigned for that SAMPLE value.

If the palette is stored in the data file, then it should be stored in BINARY format as 256 consecutive 8-bit values for each primary color (RED, GREEN, BLUE) resulting in a 768 byte record.

Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. ROW_BYTES
4. COLUMNS

Optional Keywords

1. DESCRIPTION
2. NAME

Required Objects

1. COLUMN

Optional Objects

None

Example

The examples below depict the differences between the two types of PALETTE objects. The first is an example of an ASCII PALETTE object, and the second is an example of the BINARY PALETTE object.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 80
FILE_RECORDS                    = 256
```

```
^PALETTE                        = "PALETTE.TAB"
 /* Image Palette description  */
SPACECRAFT_NAME                 = MAGELLAN
MISSION_PHASE_NAME              = PRIMARY_MISSION
TARGET_NAME                     = VENUS
PRODUCT_ID                      ="GEDR-MERC.1;2"
IMAGE_ID                        ="GEDR-MERC.1;2"
INSTRUMENT_NAME                 ="RADAR SYSTEM"
PRODUCT_CREATION_TIME           = 1995-01-01T00:00:00
NOTE                            = "Palette for browse image"

/* Description of an ASCII PALETTE object */

OBJECT                          = PALETTE
INTERCHANGE_FORMAT              = ASCII
ROWS                            = 256
ROW_BYTES                       = 80
COLUMNS                         = 4
OBJECT                          = COLUMN
NAME                            = SAMPLE
DESCRIPTION                     ="DN value for red, green, blue intensities"
DATA_TYPE                       = INTEGER
START_BYTE                      = 1
BYTES                           = 3
END_OBJECT
OBJECT                          = COLUMN
NAME                            = RED
DESCRIPTION                     = "Red intensity (0 - 255)"
DATA_TYPE                       = INTEGER
START_BYTE                      = 6
BYTES                           = 3
END_OBJECT
OBJECT                          = COLUMN
NAME                            = GREEN
DESCRIPTION                     = "Green intensity (0 - 255)"
DATA_TYPE                       = INTEGER
START_BYTE                      = 11
BYTES                           = 3
END_OBJECT
OBJECT                          = COLUMN
NAME                            = BLUE
DESCRIPTION                     = "Blue intensity (0 - 255)"
DATA_TYPE                       = INTEGER
START_BYTE                      = 16
BYTES                           = 3
END_OBJECT
END_OBJECT
END

------------------------------------------------------------

/* Description of a BINARY PALETTE object */

OBJECT                          = PALETTE
INTERCHANGE_FORMAT              = BINARY
ROWS                            = 1
ROW_BYTES                       = 768
COLUMNS                         = 3
```

```
OBJECT                          = COLUMN
NAME                            = RED
DATA_TYPE                       = UNSIGNED_INTEGER
START_BYTE                      = 1
ITEMS                           = 256
ITEM_BYTES                      = 1
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
NAME                            = GREEN
DATA_TYPE                       = UNSIGNED_INTEGER
START_BYTE                      = 257
ITEMS                           = 256
ITEM_BYTES                      = 1
END_OBJECT                      = COLUMN

OBJECT                          = COLUMN
NAME                            = BLUE
DATA_TYPE                       = UNSIGNED_INTEGER
START_BYTE                      = 513
ITEMS                           = 256
ITEM_BYTES                      = 1
END_OBJECT                      = COLUMN
END_OBJECT                      = PALETTE
END
```

A.23 QUBE

A generalized QUBE object is a multidimensional array (called the core) of sample values in multiple dimensions. The core is homogeneous, and consists of unsigned byte, signed halfword or floating point fullword elements. QUBEs of one to three dimensions may have optional suffix areas in each axis. The suffix areas may be heterogeneous, with elements of different types, but each suffix pixel is always allocated a fullword. Special values may be defined for the core and the suffix areas to designate missing values and several kinds of invalid values, such as instrument and representation saturation.

The QUBE is the principal data structure of the ISIS (Integrated Software for Imaging Spectrometers) system. A frequently used specialization of the QUBE object is the ISIS Standard Qube, which is a three-dimensional QUBE with two spatial dimensions and one spectral dimension. Its axes have the interpretations 'sample', 'line' and 'band'. Three physical storage orders are allowed: band-sequential, line_interleaved (band-interleaved-by-line) and sample_interleaved (band-interleaved-by-pixel).

An example of a Standard ISIS Qube is a spectral image qube containing data from an imaging spectrometer. Such a qube is simultaneously a set of images (at different wavelengths) of the same target area, and a set of spectra at each point of the target area. Typically, suffix areas in such a qube are confined to 'backplanes' containing geometric or quality information about individual spectra, i.e. about the set of corresponding values at the same pixel location in each band.

The following diagram illustrates the general structure of a Standard ISIS Qube. Note that this is a conceptual or “logical” view of the qube.

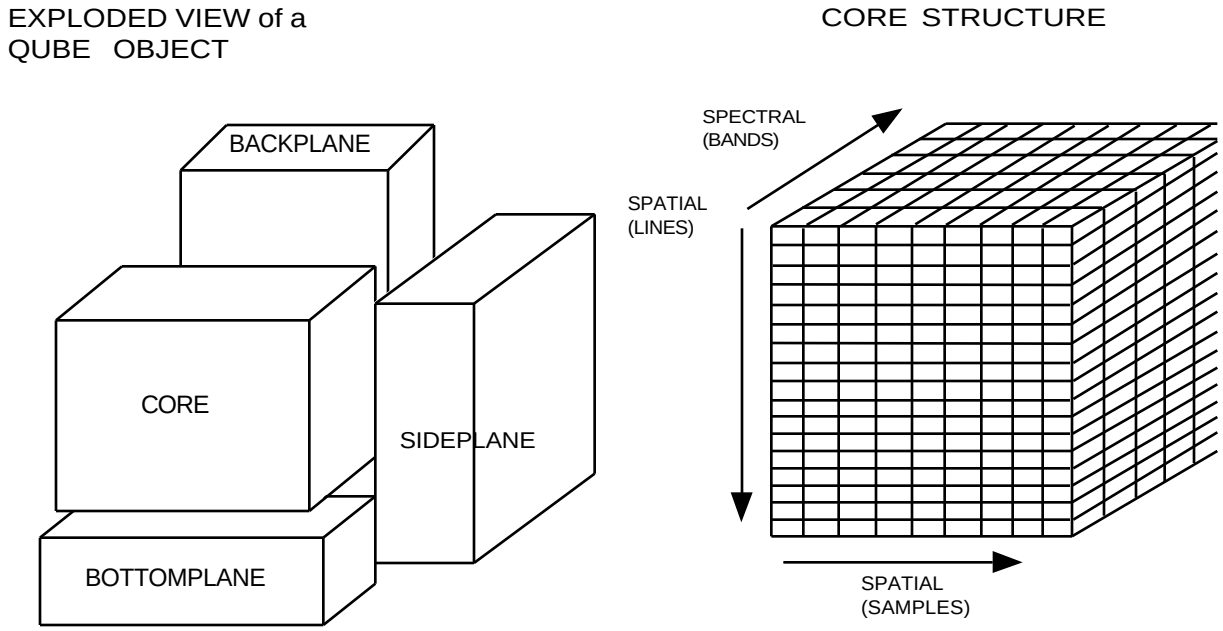


Figure A.3: Exploded View of a Qube Object

Some special requirements are imposed by the ISIS system. A QUBE object must be associated with a HISTORY object. (Other objects, such as HISTOGRAMs, IMAGEs, PALETTEs and TABLEs which contain statistics, display parameters, engineering values or other ancillary data, are optional.) A special element, FILE_STATE, is required in the implicit FILE object. Some label information is organized into GROUPs, such as BAND_BIN and IMAGE_MAP_PROJECTION. The BAND_BIN group contains essential wavelength information, and is required for Standard ISIS Qubes.

The ISIS system includes routines for reading and writing files containing QUBE objects. Both 'logical' access, independent of actual storage order, and direct 'physical' access are provided for Standard ISIS Qubes. Only physical access is provided for generalized QUBEs. Most ISIS application programs operate on Standard ISIS Qubes. Arbitrary subqubes ('virtual' qubes) of existing qubes may be specified for most of these programs. In addition, ISIS includes software for handling Tables (an ISIS variant of the PDS Table object) and Instrument Spectral Libraries.

For a complete description, refer to the most recent version of 'ISD: ISIS System Design, Build 2', obtainable from the PDS Operator.

NOTE: The following required and optional elements of the QUBE object are ISIS-specific. Since the ISIS system was designed before the current version of the Planetary Science Data Dictionary, some of the element names conflict with current PDS nomenclature standards.

Required Keywords (Generalized Qube and Standard ISIS Qube)

AXES	Number of axes or dimensions of qube [integer]
AXIS_NAME	Names of axes [sequence of 1-6 literals] (BAND, LINE, SAMPLE) for Standard Qube
CORE_ITEMS	Core dimensions of axes [seq of 1-6 integers]
CORE_ITEM_BYTES	Core element size [integer bytes: {1, 2, 4}]
CORE_ITEM_TYPE	Core element type [literal: {UNSIGNED_INTEGER, INTEGER, REAL}]
CORE_BASE	Base value of core item scaling [real]
CORE_MULTIPLIER	Multiplier for core item scaling [real] 'true' value = base + multiplier * 'stored' value (base = 0.0 and multiplier = 1.0 for REALs)
SUFFIX_BYTES	Storage allocation of suffix elements [integer: always 4]
SUFFIX_ITEMS	Suffix dimensions of axes [seq of 1-6 integers]
CORE_VALID_MINIMUM	Minimum valid core value -- values below this value are reserved for 'special' values, of which 5 are currently assigned [integer or non-decimal integer: these values are fixed by ISIS convention for each allowable item type and size -- see ISD for

	details]
CORE_NULL	Special value indicating 'invalid' data
CORE_LOW_INSTR_SATURATION	Special value indicating instrument saturation at the low end
CORE_HIGH_INSTR_SATURATION	Special value indicating instrument saturation at the high end
CORE_LOW_REPR_SATURATION	Special value indicating representation saturation at the low end
CORE_HIGH_REPR_SATURATION	Special value indicating representation saturation at the high end

Required Keywords (Standard ISIS Qube) and Optional Keywords (Generalized Qube)

CORE_NAME	Name of value stored in core of qube [literal, e.g. SPECTRAL_RADIANCE]
CORE_UNIT	Unit of value stored in core of qube [literal]
BAND_BIN_CENTER	Wavelengths of bands in a Standard Qube [sequence of reals]
BAND_BIN_UNIT	Unit of wavelength [literal, e.g. MICROMETER]
BAND_BIN_ORIGINAL_BAND	Original band numbers, referring to a Qube of which the current qube is a subqube. In the original qube, these are sequential integers.[sequence of integers]

Optional Keywords (Generalized Qube and Standard ISIS Qube)

BAND_BIN_WIDTH	Width (at half height) of spectral response of bands [sequence of reals]
BAND_BIN_STANDARD_DEVIATION	Standard deviation of spectrometer values at each band [sequence of reals]
BAND_BIN_DETECTOR	Instrument detector number of band, where relevant [sequence of integers]
BAND_BIN_GRATING_POSITION	Instrument grating position of band, where relevant [sequence of integers]

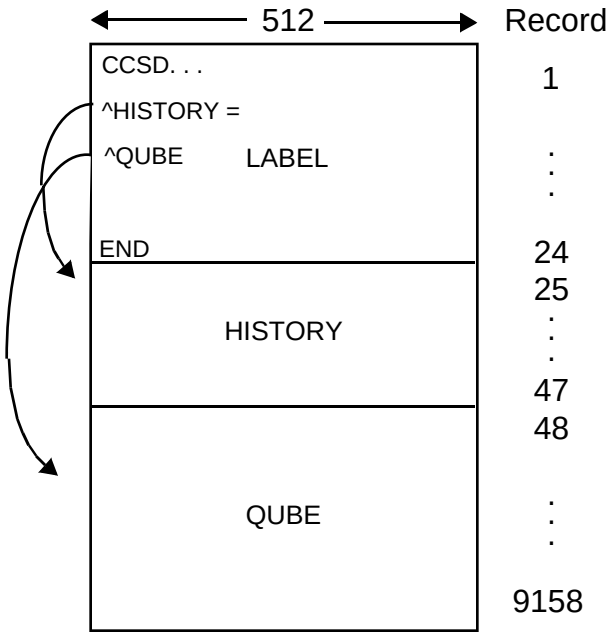
Required Keywords (for each suffix present in a 1-3 dimensional qube).
Note: These must be prefixed by the specific AXIS_NAME. These are SAMPLE, LINE and BAND for Standard ISIS Qubes. Only the commonly used BAND variants are shown:

BAND_SUFFIX_NAME	Names of suffix items [sequence of literals]
BAND_SUFFIX_UNIT	Units of suffix items [sequence of literals]
BAND_SUFFIX_ITEM_BYTES	Suffix item sizes [sequence of integer bytes {1, 2, 4}]
BAND_SUFFIX_ITEM_TYPE	Suffix item types [sequence of literals:

	{UNSIGNED_INTEGER, INTEGER, REAL, ...}]
BAND_SUFFIX_BASE	Base values of suffix item scaling [sequence of reals] (see corresponding core element)
BAND_SUFFIX_MULTIPLIER	Multipliers for suffix item scaling [sequence of reals] (see corresponding core element)
BAND_SUFFIX_VALID_MINIMUM	Minimum valid suffix values
BAND_SUFFIX_NULL	...and assigned special values
BAND_SUFFIX_LOW_INSTR_SAT	[sequences of integers or reals]
BAND_SUFFIX_HIGH_INSTR_SAT	(see corresponding core
BAND_SUFFIX_LOW_REPR_SAT	element definitions for
BAND_SUFFIX_HIGH_REPR_SAT	details)

Example

The following label describes ISIS qube data from the Galileo NIMS experiment. The qube contains 17 bands of NIMS fixed-map mode raw data numbers and 9 backplanes of ancillary information. In other modes, NIMS can produce data qubes of 34, 102, 204 and 408 bands.




```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID = PDS3
/* File Structure */
```

```
RECORD_TYPE                         = FIXED_LENGTH
RECORD_BYTES                        = 512
FILE_RECORDS                        =   9158
LABEL_RECORDS                       =     24
FILE_STATE                          = CLEAN

^HISTORY                            =     25
OBJECT                              = HISTORY
END_OBJECT                          = HISTORY

^QUBE                               =    48
OBJECT                              = QUBE

/*  Qube structure: Standard ISIS Cube of NIMS Data */

 AXES                               = 3
 AXIS_NAME                          = (SAMPLE,LINE,BAND)

/*  Core description */

 CORE_ITEMS                         = (229,291,17)
 CORE_ITEM_BYTES                    = 2
 CORE_ITEM_TYPE                     = VAX_INTEGER
 CORE_BASE                          = 0.0
 CORE_MULTIPLIER                    = 1.0
 CORE_VALID_MINIMUM                 = -32752
 CORE_NULL                          = -32768
 CORE_LOW_REPR_SATURATION           = -32767
 CORE_LOW_INSTR_SATURATION          = -32766
 CORE_HIGH_INSTR_SATURATION         = -32765
 CORE_HIGH_REPR_SATURATION          = -32764
 CORE_NAME                          = RAW_DATA_NUMBER
 CORE_UNIT                          = DIMENSIONLESS

 PHOTOMETRIC_CORRECTION_TYPE        = NONE

/*  Suffix description  */

 SUFFIX_BYTES                       = 4
 SUFFIX_ITEMS                       = (0,0,9)

 BAND_SUFFIX_NAME                   = (LATITUDE,LONGITUDE,INCIDENCE_ANGLE,
  EMISSION_ANGLE,PHASE_ANGLE,SLANT_DISTANCE,INTERCEPT_ALTITUDE,
  PHASE_ANGLE_STD_DEV,RAW_DATA_NUMBER_STD_DEV)
 BAND_SUFFIX_UNIT                   = (DEGREE,DEGREE,DEGREE,DEGREE,DEGREE,KILOMETER,
  KILOMETER,DEGREE,DIMENSIONLESS)
 BAND_SUFFIX_ITEM_BYTES             = (4,4,4,4,4,4,4,4,4)
 BAND_SUFFIX_ITEM_TYPE              = (VAX_REAL,VAX_REAL,VAX_REAL,VAX_REAL,VAX_REAL,
    VAX_REAL,VAX_REAL,VAX_REAL,VAX_REAL)
 BAND_SUFFIX_BASE                   = (0.000000,0.000000,0.000000,0.000000,0.000000,
    0.000000,0.000000,0.000000,0.000000)
 BAND_SUFFIX_MULTIPLIER             = (1.000000,1.000000,1.000000,1.000000,1.000000,
    1.000000,1.000000,1.000000,1.000000)
 BAND_SUFFIX_VALID_MINIMUM          = (16#FFEFFFFF#,16#FFEFFFFF#,16#FFEFFFFF#,
    16#FFEFFFFF#,16#FFEFFFFF#,16#FFEFFFFF#,16#FFEFFFFF#,16#FFEFFFFF#,
    16#FFEFFFFF#)
 BAND_SUFFIX_NULL                   = (16#FFFFFFFF#,16#FFFFFFFF#,16#FFFFFFFF#,16#FFFFFFFF#,
    16#FFFFFFFF#,16#FFFFFFFF#,16#FFFFFFFF#,16#FFFFFFFF#,16#FFFFFFFF#)
```

```
 BAND_SUFFIX_LOW_REPR_SAT              = (16#FFFEFFFF#,16#FFFEFFFF#,16#FFFEFFFF#,
    16#FFFEFFFF#,16#FFFEFFFF#,16#FFFEFFFF#,16#FFFEFFFF#,16#FFFEFFFF#,
    16#FFFEFFFF#)
 BAND_SUFFIX_LOW_INSTR_SAT             = (16#FFFDFFFF#,16#FFFDFFFF#,16#FFFDFFFF#,
    16#FFFDFFFF#,16#FFFDFFFF#,16#FFFDFFFF#,16#FFFDFFFF#,16#FFFDFFFF#,
    16#FFFDFFFF#)
 BAND_SUFFIX_HIGH_INSTR_SAT            = (16#FFFCFFFF#,16#FFFCFFFF#,16#FFFCFFFF#,
    16#FFFCFFFF#,16#FFFCFFFF#,16#FFFCFFFF#,16#FFFCFFFF#,16#FFFCFFFF#,
    16#FFFCFFFF#)
 BAND_SUFFIX_HIGH_REPR_SAT             = (16#FFFBFFFF#,16#FFFBFFFF#,16#FFFBFFFF#,
    16#FFFBFFFF#,16#FFFBFFFF#,16#FFFBFFFF#,16#FFFBFFFF#,16#FFFBFFFF#,
    16#FFFBFFFF#)
 BAND_SUFFIX_NOTE                      = "
```

The backplanes contain 7 geometric parameters, the standard deviation of one of them, the standard deviation of a selected data band, and 0 to 10 'spectral index' bands, each a user-specified function of the data bands. (See the BAND_SUFFIX_NAME values.)

Longitude ranges from 0 to 360 degrees, with positive direction specified by POSITIVE_LONGITUDE_DIRECTION in the IMAGE_MAP_PROJECTION group.

INTERCEPT_ALTITUDE contains values for the DIFFERENCE between the length of the normal from the center of the target body to the line of sight AND the radius of the target body. On-target points have zero values. Points beyond the maximum expanded radius have null values. This plane thus also serves as a set of 'off-limb' flags. It is meaningful only for the ORTHOGRAPHIC and POINT_PERSPECTIVE projections; otherwise all values are zero. The geometric standard deviation backplane contains the standard deviation of the geometry backplane indicated in its NAME, except that the special value 16#FFF9FFFF# replaces the standard deviation where the corresponding core pixels have been 'filled'.

The data band standard deviation plane is computed for the NIMS data band specified by STD_DEV_SELECTED_BAND_NUMBER. This may be either a raw data number, or spectral radiance, whichever is indicated by CORE_NAME.

The (optional) spectral index bands were generated by the Vicar F2 program. The corresponding BAND_SUFFIX_NAME is an abbreviated formula for the function used, where Bn should be read 'NIMS data band n'. For example: B4/B8 represents the ratio of bands 4 and 8."

```
STD_DEV_SELECTED_BAND_NUMBER       = 9

/*  Data description: general */

DATA_SET_ID                        = "GO-V-NIMS-4-MOSAIC-V1.0"
PRODUCT_ID                         = "XYZ"
SPACECRAFT_NAME                    = GALILEO_ORBITER
MISSION_PHASE_NAME                 = VENUS_ENCOUNTER
INSTRUMENT_NAME                    = NEAR_INFRARED_MAPPING_SPECTROMETER
INSTRUMENT_ID                      = NIMS
^INSTRUMENT_DESCRIPTION            = "NIMSINST.TXT"

TARGET_NAME                        = VENUS
START_TIME                         = 1990-02-10T01:49:58Z
STOP_TIME                          = 1990-02-10T02:31:52Z
NATIVE_START_TIME                  =   180425.85
NATIVE_STOP_TIME                   =   180467.34
OBSERVATION_NAME                   = 'VPDIN1'
OBSERVATION_NOTE                   = "VPDIN1 / Footprint, Limbfit, Height=50"

INCIDENCE_ANGLE                    = 160.48
EMISSION_ANGLE                     = 14.01
PHASE_ANGLE                        = 147.39
SUB_SOLAR_AZIMUTH                  = -174.74
```

```
SUB_SPACECRAFT_AZIMUTH              = -0.80
MINIMUM_SLANT_DISTANCE              =  85684.10
MAXIMUM_SLANT_DISTANCE              = 103175.00
MIN_SPACECRAFT_SOLAR_DISTANCE       = 1.076102e+08
MAX_SPACECRAFT_SOLAR_DISTANCE       = 1.076250e+08

/*  Data description: instrument status  */

INSTRUMENT_MODE_ID                  = FIXED_MAP
GAIN_MODE_ID                        = 2
CHOPPER_MODE_ID                     = REFERENCE
START_GRATING_POSITION              = 16
OFFSET_GRATING_POSITION             = 04

MEAN_FOCAL_PLANE_TEMPERATURE        = 85.569702
MEAN_RAD_SHIELD_TEMPERATURE         = 123.636002
MEAN_TELESCOPE_TEMPERATURE          = 139.604996
MEAN_GRATING_TEMPERATURE            = 142.580002
MEAN_CHOPPER_TEMPERATURE            = 142.449997
MEAN_ELECTRONICS_TEMPERATURE        = 287.049988

GROUP                               = BAND_BIN

/*  Spectral axis description */

BAND_BIN_CENTER                     = (0.798777,0.937873,1.179840,1.458040,1.736630,
   2.017250,2.298800,2.579060,2.864540,3.144230,3.427810,3.710640,
   3.993880,4.277290,4.561400,4.843560,5.126080)
BAND_BIN_UNIT                       = MICROMETER
BAND_BIN_ORIGINAL_BAND              = (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17)
BAND_BIN_GRATING_POSITION           = (16,16,16,16,16,16,16,16,16,16,16,16,
16,16,16,16,16)
BAND_BIN_DETECTOR                   = (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17)
END_GROUP                           = BAND_BIN

GROUP                               = IMAGE_MAP_PROJECTION
/* Projection description */
MAP_PROJECTION_TYPE                 = OBLIQUE_ORTHOGRAPHIC
MAP_SCALE                           = 45.000
MAP_RESOLUTION                      =  2.366
CENTER_LATITUDE                     =  12.00
CENTER_LONGITUDE                    = 350.00
LINE_PROJECTION_OFFSET              = 149.10
SAMPLE_PROJECTION_OFFSET            =  85.10
MINIMUM_LATITUDE                    =  11.71
MAXIMUM_LATITUDE                    =  13.62
MINIMUM_LONGITUDE                   = 349.62
MAXIMUM_LONGITUDE                   = 351.72
POSITIVE_LONGITUDE_DIRECTION        = EAST
A_AXIS_RADIUS                       = 6101.000000
B_AXIS_RADIUS                       = 6101.000000
C_AXIS_RADIUS                       = 6101.000000
REFERENCE_LATITUDE                  = 0.000000
REFERENCE_LONGITUDE                 = 0.000000
MAP_PROJECTION_ROTATION             =   0.00
LINE_FIRST_PIXEL                    = 1
LINE_LAST_PIXEL                     = 229
SAMPLE_FIRST_PIXEL                  = 1
```

```
SAMPLE_LAST_PIXEL                = 291
END_GROUP                        = IMAGE_MAP_PROJECTION

END_OBJECT                       = QUBE
END
```

A.24 SERIES

The SERIES object is a sub-class of the TABLE object. It is used for storing a sequence of measurements organized in a specific way (e.g. ascending time, radial distances). The current version uses the same physical format specification as the TABLE object, but includes sampling parameter information that describes the variation between elements in the series.

The sampling parameter keywords are required for the SERIES object and may be optional for one or more COLUMN sub-objects, depending on the data organization.

The sampling parameter keywords in the SERIES object represent the variation between the ROWS of data. For data that vary regularly between each row, the SAMPLING_PARAMETER_INTERVAL keyword defines this regularity. For data in which rows are irregularly spaced, the SAMPLING_PARAMETER_INTERVAL keyword is “N/A”, and the actual sampling parameter values are included in the data itself and identified as a column in the series. An example of this is a file of time series data with rows ordered by a time column (or set of columns).

For data that vary regularly between items of a single column, sampling parameter keywords appear as part of the COLUMN sub-object. Data sampled at irregular intervals described as separate columns may also provide sampling parameter information specific to each column.

Optional MINIMUM_SAMPLING_PARAMETER and MAXIMUM_SAMPLING_PARAMETER keywords should be added whenever possible to indicate the range in which the data was sampled. For data sampled at a single point rather than over a range, both the MINIMUM_SAMPLING_PARAMETER and MAXIMUM_SAMPLING_PARAMETER are set to the specific value. For TIME_SERIES data, where the sampling parameter specified is time, these keywords are not used.

Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES
5. SAMPLING_PARAMETER_NAME
6. SAMPLING_PARAMETER_UNIT
7. SAMPLING_PARAMETER_INTERVAL

Optional Keywords

1. NAME
2. ROW_PREFIX_BYTES
3. ROW_SUFFIX_BYTES
4. MINIMUM_SAMPLING_PARAMETER
5. MAXIMUM_SAMPLING_PARAMETER

6. DERIVED_MINIMUM
7. DERIVED_MAXIMUM
8. DESCRIPTION

Required Objects

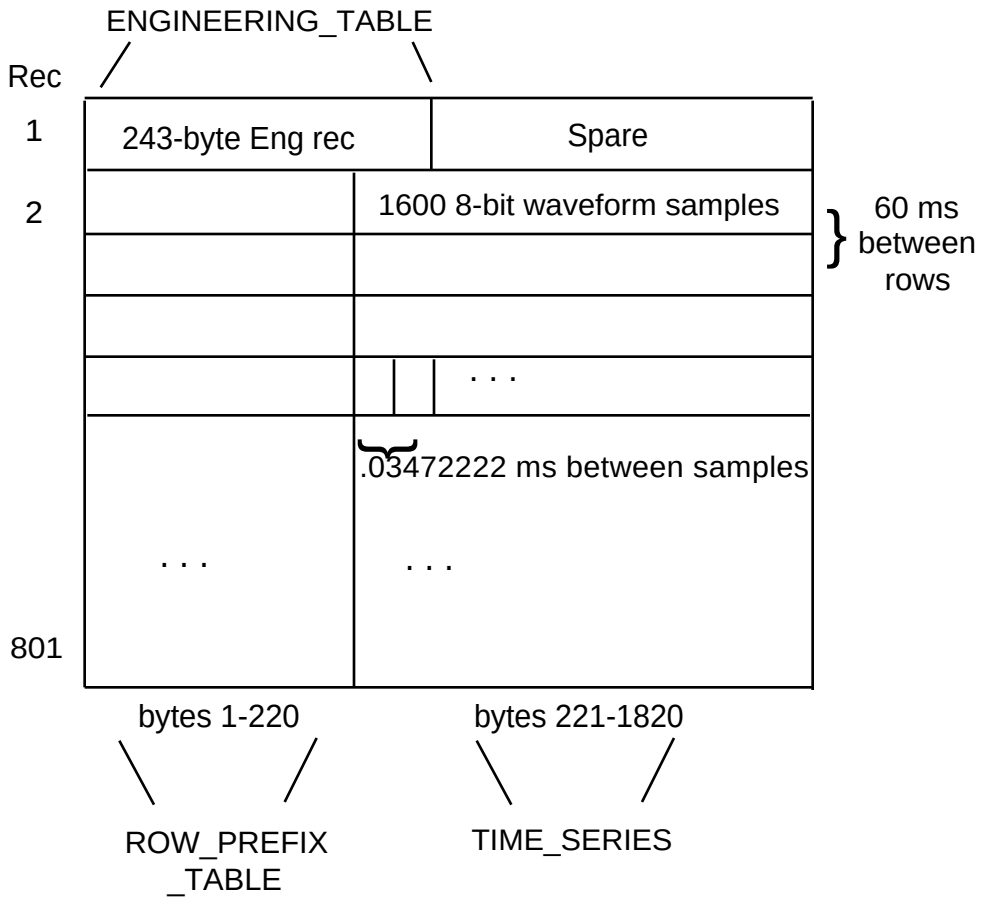
1. COLUMN

Optional Objects

1. CONTAINER

Example

This example illustrates the use of the SERIES object for data that vary regularly in two ways. Rows of data in the SERIES occur at 60 millisecond intervals while the COLUMN occurs at .03472222 millisecond intervals.



```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3
RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 1820
FILE_RECORDS                            = 801
^ENGINEERING_TABLE                      = ("C0900313.DAT", 1)
^ROW_PREFIX_TABLE                       = ("C0900313.DAT", 2)
^TIME_SERIES                            = ("C0900313.DAT", 2)
/* Observation description */
DATA_SET_ID                             = "VG2-N-PWS-2-EDR-WFRM-60MS-V1.0"
PRODUCT_ID                              = "C0900313.DAT"
PRODUCT_CREATION_TIME                   = "UNK"
SPACECRAFT_NAME                         = VOYAGER_2
SPACECRAFT_CLOCK_START_COUNT            = "09003.13.002"
SPACECRAFT_CLOCK_STOP_COUNT             = "09003.13.002"
EARTH_RECEIVED_TIME                     = 1989-159T13:35:00.121Z
START_TIME                              = 1989-157T14:16:56.979Z
STOP_TIME                               = "N/A"
MISSION_PHASE_NAME                      = NEPTUNE_ENCOUNTER
TARGET_NAME                             = NEPTUNE
/* Instrument description */
INSTRUMENT_NAME                         = PLASMA_WAVE_RECEIVER
INSTRUMENT_ID                           = PWS
SECTION_ID                              = WFRM
/* Object descriptions */
OBJECT                                  = ENGINEERING_TABLE
INTERCHANGE_FORMAT                      = BINARY
ROWS                                    = 1
COLUMNS                                 = 106
ROW_BYTES                               = 243
ROW_SUFFIX_BYTES                        =1577
DESCRIPTION                             = "This table describes the format of the engineering record which is included as
the first record in  each PWS high rate waveform file.  This record contains the  first 242 bytes of data extracted from the Mission
and Test Imaging System (MTIS) header record on each file of an imaging EDR tape.  A 243rd byte containing some flag fields has
been added to the table for all data collected during the Neptune encounter."
  ^STRUCTURE                            = "ENGTAB.FMT"
END_OBJECT                              = ENGINEERING_TABLE

OBJECT                                  = ROW_PREFIX_TABLE
INTERCHANGE_FORMAT                      = BINARY
ROWS                                    = 800
COLUMNS                                 = 47
ROW_BYTES                               = 220
ROW_SUFFIX_BYTES                        = 1600
DESCRIPTION                             = "This table describes the format of the engineering data associated with the
collection of each row of waveform data (1600 waveform samples)."
^STRUCTURE                              = "ROWPRX.FMT"
END_OBJECT                              = ROW_PREFIX_TABLE

OBJECT                                  = TIME_SERIES
NAME                                    = WAVEFORM_FRAME
INTERCHANGE_FORMAT                      = BINARY
ROWS                                    = 799
COLUMNS                                 = 1
ROW_BYTES                               = 1600
ROW_PREFIX_BYTES                        = 220
SAMPLING_PARAMETER_NAME                 = TIME
```

SAMPLING_PARAMETER_UNIT = SECOND
SAMPLING_PARAMETER_INTERVAL = .06 /* 60 MS between rows */
DESCRIPTION = "This time_series consists of up to 800 records (or rows, lines) of PWS waveform sample data. Each record 2-801 of the file (or frame) contains 1600 waveform samples, prefaced by 220 bytes of MTIS information. The 1600 samples are collected in 55.56 msec followed by a 4.44 msec gap. Each 60 msec interval constitutes a line of waveform samples. Each file contains up to 800 lines of waveform samples for a 48 sec frame."

OBJECT = COLUMN
NAME = WAVEFORM_SAMPLES
DATA_TYPE = MSB_UNSIGNED_INTEGER
START_BYTE = 221
BYTES = 1600
ITEMS = 1600
ITEM_BYTES = 1
SAMPLING_PARAMETER_NAME = TIME
SAMPLING_PARAMETER_UNIT = SECOND
SAMPLING_PARAMETER_INTERVAL = 0.00003472222 /* time between samples */

OFFSET = -7.5
VALID_MINIMUM = 0
VALID_MAXIMUM = 15
DESCRIPTION = "The 1 byte waveform samples constitute an array of waveform measurements which are encoded into binary values from 0 to 15 and may be re-mapped to reduce the artificial zero-frequency component. For example, stored values can be mapped to the following floating point values. The original 4-bit data samples have been repackaged into 8-bit (1 byte) items without modification for archival purposes.\n

0 = -7.5	1 = -6.5	2 = -5.5	3 = -4.5
4 = -3.5	5 = -2.5	6 = -1.5	7 = -0.5
8 =0.5	9 =1.5	10=2.5	11 =3.5
12 =4.5	13 =5.5	14 =6.5	15 =7.5

"
END_OBJECT = COLUMN
END_OBJECT = TIME_SERIES

END

A.25 SPECTRUM

The SPECTRUM object is a form of TABLE used for storing spectral measurements. The SPECTRUM object is assumed to have a number of measurements of the observation target taken in different SPECTRAL bands. The SPECTRUM object uses the same physical format specification as the TABLE object, but includes a SAMPLING PARAMETER definition which indicates the spectral region measured in successive COLUMNs or ROWs. The common sampling parameters for SPECTRUM objects are wavelength, frequency, or velocity.

A regularly sampled SPECTRUM can be stored either horizontally as a 1 row table with 1 column containing n samples (expressed as ITEMS=n), or vertically as a 1 column table with n rows where each ROW contains a sample of the spectrum. The vertical format allows additional columns to be defined for related parameters for each sample value (e.g. ERROR factors). These related columns can be described in a separate PREFIX or SUFFIX table.

An irregularly sampled SPECTRUM must be stored horizontally, with each specific spectral range identified as a separate column, and defined by a specific set of sampling parameter keywords for each column.

In the horizontal format, the sampling parameter specifications are included in the COLUMN definition. For a vertically defined SPECTRUM, the sampling parameter information is provided in the SPECTRUM object, since it is describing the spectral variation between the ROWs of the data.

Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES

Optional Keywords

1. NAME
2. SAMPLING_PARAMETER_NAME
3. SAMPLING_PARAMETER_UNIT
4. SAMPLING_PARAMETER_INTERVAL
5. ROW_PREFIX_BYTES
6. ROW_SUFFIX_BYTES
7. MINIMUM_SAMPLING_PARAMETER
8. MAXIMUM_SAMPLING_PARAMETER
9. DERIVED_MINIMUM
10. DERIVED_MAXIMUM
11. DESCRIPTION

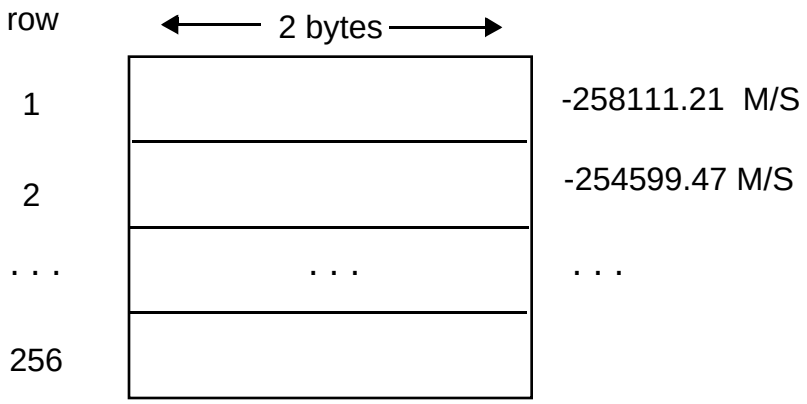
Required Objects

1. COLUMN

Optional Objects

1. CONTAINER

Example
This example illustrates a SPECTRUM data object stored in a vertical format. The data are regularly sampled at intervals of 99.09618 meters/second and data samples are stored in successive ROWS.




```
CCSD3ZF0000100000001NJPL3IFOPDSX00000001
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 2
FILE_RECORDS                    = 256
PRODUCT_ID                      = "RSSL007.DAT"
DATA_SET_ID                     =  "IHW-C-RSSL-3-EDR-HALLEY-V1.0"
TARGET_NAME                     =  "HALLEY"
INSTRUMENT_HOST_NAME            = "IHW RADIO STUDIES NETWORK"
INSTRUMENT_NAME                 = "RADIO SPECTRAL LINE DATA"
OBSERVATION_ID                  = "621270"
START_TIME                      = 1985-11-10T00:43:12.000
STOP_TIME                       = 1985-11-10T00:43:12.000
PRODUCT_CREATION_TIME           = "UNK"
/* Record Pointer to Major Object */
^TOTAL_INTENSITY_SPECTRUM       = "RSSL0007.DAT"
/* Object Description */

OBJECT                          = SPECTRUM
INTERCHANGE_FORMAT              = BINARY
ROWS                            = 256
ROW_BYTES                       = 2
COLUMNS                         = 1
SAMPLING_PARAMETER_NAME         = "VELO_COM"
```

```
MINIMUM_SAMPLING_PARAMETER      = -1.268431E+04
SAMPLING_PARAMETER_INTERVAL     = 9.909618E+01
SAMPLING_PARAMETER_UNIT         = "METERS/SECOND"
DESCRIPTION                     = "Radio Studies; Spectral Line intensity spectrum.  Spectrum is organized as 1
column with 256 rows.  Each row contains a spectral value for the velocity derived from the sampling parameter information
associated with each row."

OBJECT                          = COLUMN
NAME                            = FLUX_DENSITY
DATA_TYPE                       = MSB_INTEGER
START_BYTE                      = 1
BYTES                           = 2
SCALING_FACTOR                  = 7.251200E-04
OFFSET                          = 0.000000E+01
DERIVED_MINIMUM                 = 2.380000E+01
DERIVED_MAXIMUM                 = 3.490000E+01
END_OBJECT                      = COLUMN
END_OBJECT                      = SPECTRUM

END
```

A.26 SPICE KERNEL

The SPICE_KERNEL object defines a single kernel (file) from a collection of SPICE Kernels. SPICE kernels provide ancillary data needed to support the planning and subsequent analysis of space science observations.

The SPICE system includes the software and documentation required to read the SPICE Kernels and use the data contained therein to help plan observations or interpret space science data. This software and associated documentation are collectively called the NAIF Toolkit.

Kernel files are the major components of the SPICE system. The EPHEMERIS KERNEL_TYPE (SPK) contains spacecraft and planet, satellite or other target body ephemeris data that provide position and velocity of a spacecraft as a function of time. The TARGET_CONSTANTS KERNEL_TYPE (PCK) contains planet, satellite, comet, or asteroid cartographic constants for that object. The INSTRUMENT KERNEL_TYPE (IK) contains a collection of science instrument information, including specification of the mounting alignment, internal timing, and other information needed to interpret measurements made with the instrument. The POINTING KERNEL_TYPE (CK) contains pointing data (e.g., the inertially referenced attitude for a spacecraft structure upon which instruments are mounted, given as a function of time). The EVENTS KERNEL_TYPE (EK) contains event information (e.g, spacecraft and instrument commands, ground data system event logs, and experimenter's notebook comments). The LEAPSECONDS KERNEL_TYPE (LSK) contains an account of the leapseconds needed to correlate civil time (UTC or GMT) with ephemeris time (TDB). This is the measure of time used in the SP kernel files. The Spacecraft Clock coefficients kernel (SCLK) contains the data needed to correlate a spacecraft clock with ephemeris time.

Data products referencing a particular SPICE kernel would do so through the SOURCE_PRODUCT_ID keyword in their label with the value corresponding to that of the PRODUCT_ID within the SPICE_KERNEL label. The PRODUCT_ID keyword is unique to a data product.

Required Keywords

1. DESCRIPTION
2. INTERCHANGE_FORMAT
3. KERNEL_TYPE

Optional Keywords

None

Required Objects

None

Optional Objects

None

Example

NOTE: The following example of a SPICE CK (Pointing) Kernel attached label may have been modified to reflect current PDS standards and is not intended to contain actual PDS ingested values. You will notice that some label information is actually inside the Kernel file which allows NAIF tools to extract information to produce the PDS label.

CCSD...
PDS_VERSION_ID = PDS3
RECORD_TYPE = STREAM
MISSION_NAME = MARS_OBSERVER
SPACECRAFT_NAME = MARS_OBSERVER
DATA_SET_ID = “MO-M-SPICE-6-CK-V1.0”
FILE_NAME = "NAF0000D.TC"
PRODUCT_ID = "NAF0000D-CK"
PRODUCT_CREATION_TIME = 1992-04-14T12:00:00
PRODUCER_ID = "NAIF"
MISSION_PHASE_TYPE = "ORBIT"
PRODUCT_VERSION_TYPE = "TEST"
START_TIME = 1994-01-06T00:00:00
STOP_TIME = 1994-02-04T23:55:00
SPACECRAFT_CLOCK_START_COUNT = "3/76681108.213"
SPACECRAFT_CLOCK_STOP_COUNT = "4/79373491.118"
TARGET_NAME = MARS
INSTRUMENT_NAME = "MARS OBSERVER SPACECRAFT"
INSTRUMENT_ID = MO
SOURCE_PRODUCT_ID =
{"NAF0000C.BSP","NAF0000C.TLS","NAF0000C.TSC"}
NOTE = "BASED ON EPHEMERIS IN NAF0000C.BSP. FOR SOFTWARE
TESTING ONLY."
OBJECT = SPICE_KERNEL
INTERCHANGE_FORMAT = ASCII
KERNEL_TYPE = POINTING
DESCRIPTION = "This is a SPICE kernel file, designed to be accessed using NAIF Toolkit software. Contact your flight project representative or the NAIF node of the Planetary Data System if you wish to obtain a copy of the NAIF Toolkit. The Toolkit consists of portable FORTRAN 77 code and extensive user documentation."
END_OBJECT = SPICE_KERNEL
END
CCSD...

INTERNAL SPICE LABEL
SPICE DATA

A.27 TABLE

TABLEs are the natural storage format for collections of data from many instruments. They are also the most effective way of storing much of the meta-data which are used to identify and describe instrument observations.

The TABLE object is a uniform collection of rows containing ASCII or binary values stored in columns. The ROWS and COLUMNS of the TABLE object provide a natural correspondence to the records and fields often defined in interface specifications for existing data products. The value to use for the COLUMNS keyword in a TABLE object should be the actual number of COLUMN objects defined in the label. The INTERCHANGE_FORMAT keyword is used to distinguish between ASCII and binary table values.

ASCII vs. BINARY formats

ASCII tables provide the most portable format for access across a wide variety of computer platforms. They are also easily imported into a number of database management systems and spreadsheet applications. For these reasons, the PDS recommends the use of ASCII table formats whenever possible for archive products.

ASCII formats are generally less efficient for storing large quantities of data. In addition, raw or minimally processed data products and many pre-existing data products undergoing restoration are only available in binary formats.Where conversion to an ASCII format is neither cost effective nor desirable, BINARY table formats can be used.

Required Keywords

1. INTERCHANGE_FORMAT
2. ROWS
3. COLUMNS
4. ROW_BYTES

Optional Keywords

1. NAME
2. DESCRIPTION
3. ROW_PREFIX_BYTES
4. ROW_SUFFIX_BYTES
5. TABLE_STORAGE_TYPE

Required Objects

1. COLUMN

Optional Objects

1. CONTAINER

Many variations of the TABLE object are possible with the addition of the “optional” keywords and/or objects to the basic TABLE definition. While PDS supports these options, they are often not the best choices for archival data products. Recommended ASCII and binary table formats are provided in the following sections (A.27.1, A.27.2) with examples. Section A.27.3 provides examples of several TABLE variations and their possible application. Section A.27.4 provides specific guidelines for SPARE columns or unused fields within a TABLE.

A.27.1 Recommended ASCII TABLE Format

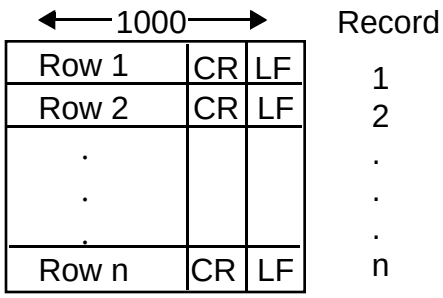
The recommended PDS table format uses ASCII COLUMN values, with a fixed size for each COLUMN. Each RECORD within the table is the same length and is terminated with a carriage-return/line-feed <CR><LF> pair. COLUMNs are separated by commas and character fields are enclosed in QUOTATION MARKS ("). The QUOTATION MARKs should surround the maximum COLUMN width. For example, a twelve character COLUMN called SPACECRAFT_NAME would be represented in the table as:

"VOYAGER 1 ", instead of "VOYAGER 1"

Numeric fields are right-justified in the allotted space and character fields are left-justified and blank padded on the right. This table format can be imported into many data management systems such as DBASE, FoxBase, Paradox, and Britton-Lee and into EXCEL spreadsheets.

The following label subset and illustration provide the general characteristics of a PDS recommended ASCII table with 1000 byte records:

```
RECORD_TYPE  = FIXED_LENGTH
RECORD_BYTES = 1000
...
OBJECT = TABLE
  INTERCHANGE_FORMAT = ASCII
  ROW_BYTES = 1000
  ...
END_OBJECT = TABLE
```



Example - Recommended ASCII TABLE

The following example is an ASCII index table with fixed length 71 byte records. Note that for ASCII tables, the delimiters (i.e., double quotes, commas, and line terminators <CR><LF>) are included in the byte count for each record (RECORD_BYTES). In this example, the delimiters are also included in the byte count for each row (ROW_BYTES). The <CR><LF> characters have been placed in columns 70 and 71.

Contents of file "INDEX.TAB"

```
-------------------------------------------------------------------------
"F-MIDR ","F-MIDR.40N286;1 ","C", 42, 37,289,282,"F40N286/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.20N280;1 ","C", 22, 17,283,277,"F20N280/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.20N286;1 ","C", 22, 17,289,283,"F20N286/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.00N279;1 ","R",  2, -2,281,275,"F00N279/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.05N290;1 ","C",  7,  2,292,286,"F05N290/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.05S279;1 ","R", -2, -7,281,275,"F05S279/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.10S284;1 ","C", -7,-12,287,281,"F10S284/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.10S290;1 ","R", -7,-12,292,286,"F10S290/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.15S283;1 ","R",-12,-17,286,279,"F15S283/FRAME.LBL "<CR><LF>
"F-MIDR ","F-MIDR.15S289;1 ","R",-12,-17,291,285,"F15S289/FRAME.LBL "<CR><LF>
```

Contents of file "INDEX.LBL"

```
-------------------------------------------------------------------------
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                  = PDS3
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 71
FILE_RECORDS                    = 10
^INDEX_TABLE                    = "INDEX.TAB"

DATA_SET_ID                     = "MGN-V-RDRS-5-MIDR-FULL-RES-V1.0"
VOLUME_ID                       = MG_7777
PRODUCT_ID                      = "FMIDR.XYZ"
SPACECRAFT_NAME                 = MAGELLAN
INSTRUMENT_NAME                 = "RADAR SYSTEM"
TARGET_NAME                     = VENUS
PRODUCT_CREATION_TIME           = "N/A"
MISSION_PHASE_NAME              = PRIMARY_MISSION
NOTE                            = "This table lists all MIDRs on this volume.  It also includes the latitude and
longitude range for each MIDR and the directory in which it is found."

OBJECT                          = INDEX_TABLE
INTERCHANGE_FORMAT              = ASCII
ROWS                            = 10
COLUMNS                         = 8
ROW_BYTES                       = 71
INDEX_TYPE                      = SINGLE

OBJECT                          = COLUMN
NAME                            = PRODUCT_TYPE
DESCRIPTION                     = "Magellan DMAT type code.  Possible values are F-MIDR, C1-MIDR, C2-
                                  MIDR, C3-MIDR, and P-MIDR."
DATA_TYPE                       = CHARACTER
```

START_BYTE = 2
BYTES = 7
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = PRODUCT_ID
DESCRIPTION = "Magellan DMAT name of product.
Example: F-MIDR.20N334;1"
DATA_TYPE = CHARACTER
START_BYTE = 12
BYTES = 16
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = SEAM_CORRECTION_TYPE
DESCRIPTION = "A value of C indicates that cross- track seam correction has been applied. A value of R indicates that the correction has not been applied."
DATA_TYPE = CHARACTER
START_BYTE = 31
BYTES = 1
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MAXIMUM_LATITUDE
DESCRIPTION = "Northernmost frame latitude rounded to the nearest degree."
DATA_TYPE = INTEGER
UNIT = DEGREE
START_BYTE = 34
BYTES = 3
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = MINIMUM_LATITUDE
DESCRIPTION = "Southernmost frame latitude rounded to the nearest degree."
DATA_TYPE = INTEGER
UNIT = DEGREE
START_BYTE = 38
BYTES = 3
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = EASTERNMOST_LONGITUDE
DESCRIPTION = "Easternmost frame longitude rounded to the nearest degree."
DATA_TYPE = INTEGER
UNIT = DEGREE
START_BYTE = 42
BYTES = 3
END_OBJECT = COLUMN

OBJECT = COLUMN
NAME = WESTERNMOST_LONGITUDE
DESCRIPTION = "Westernmost frame longitude rounded to the nearest degree."
DATA_TYPE = INTEGER
UNIT = DEGREE
START_BYTE = 46
BYTES = 3
END_OBJECT = COLUMN

OBJECT = COLUMN

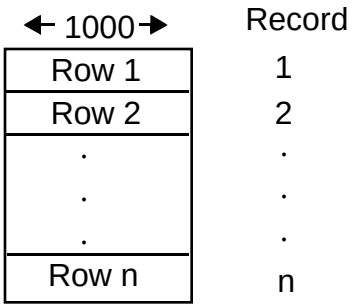
```
NAME                            = FILE_SPECIFICATION_NAME
DESCRIPTION                     = "Path and file name of frame table relative to CD-ROM root directory."
DATA_TYPE                       = CHARACTER
START_BYTE                      = 51
BYTES                           = 18
END_OBJECT                      = COLUMN

END_OBJECT                      = INDEX_TABLE
END
```

A.27.2 Recommended BINARY TABLE Format

The recommended PDS binary table format uses FIXED_LENGTH records, with each row of the table occupying a complete physical record (i.e. RECORD_BYTES = ROW_BYTES). This recommended format also discourages the use of BIT_COLUMN objects within COLUMNS in binary tables, primarily for portability reasons. Whenever possible, bit fields should be unpacked into more portable byte oriented COLUMNS. Unused bytes embedded within the binary table should be explicitly identified with COLUMNs named “SPARE” for completeness and to facilitate automated validation of these table structures.

The following label subset and illustration provide the general characteristics of a PDS recommended binary table with 1000 byte records:

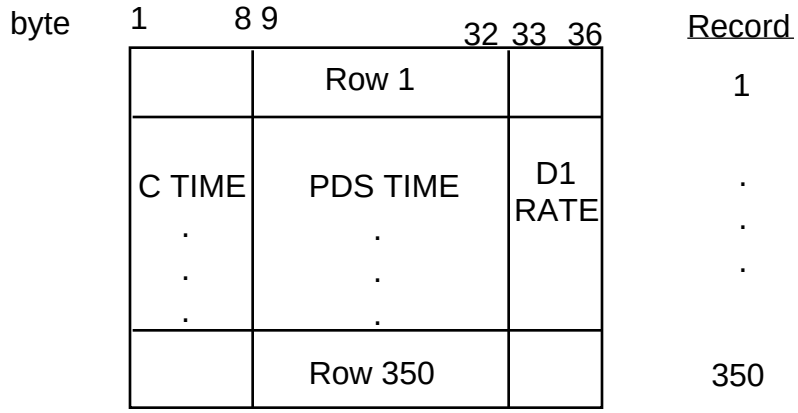
```
RECORD_TYPE  = FIXED_LENGTH
  RECORD_BYTES = 1000
  ...
  OBJECT = TABLE
    INTERCHANGE_FORMAT = BINARY
    ROW_BYTES = 1000
    ...
  END_OBJECT = TABLE
```



Example - Recommended Binary TABLE

The following is an example of a binary table consisting of 3 columns of data. The first two columns provide TIME information in both the PDS standard UTC format and an alternate format. The third column provides uncalibrated instrument measurements for the given times. This table could also be represented as a TIME_SERIES by the addition of sampling parameter keywords to describe the variation between each row of the table. The following illustration shows the layout and contents of the binary table in file "T890825.DAT". The detached label file, "T890825.LBL" provides the complete description.

Contents of file "T890825.DAT":



Contents of file "T890825.LBL":

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3

/* File Characteristic Keywords */
RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 36
FILE_RECORDS                            = 350
HARDWARE_MODEL_ID                       = "SUN SPARC STATION"
OPERATING_SYSTEM_ID                     = "SUN OS 4.1.1"

/* Data Object Pointers */
^TABLE                                  = "T890825.DAT"

/* Identification Keywords */
DATA_SET_ID                             = "VG2-N-CRS-4-SUMM-D1-96SEC-V1.0"
SPACECRAFT_NAME                         = "VOYAGER 2"
INSTRUMENT_NAME                         = "COSMIC RAY SYSTEM"
TARGET_NAME                             = NEPTUNE
START_TIME                              = 1989-08-25T00:00:00.000Z
STOP_TIME                               = 1989-08-25T09:58:02.000Z
MISSION_PHASE_NAME                      = "NEPTUNE ENCOUNTER"
PRODUCT_ID                              = "T890825.DAT"
PRODUCT_CREATION_TIME                   = "UNK"
SPACECRAFT_CLOCK_START_COUNT            = "UNK"
SPACECRAFT_CLOCK_STOP_COUNT             = "UNK"

/* Data Object Descriptions */
OBJECT                                  = TABLE
INTERCHANGE_FORMAT                      = BINARY
ROWS                                    = 350
COLUMNS                                 = 3
ROW_BYTES                               = 36
^STRUCTURE                              = "CRSDATA.FMT"
END_OBJECT                              = TABLE
END
```

Contents of file "CRSDATA.FMT":

```
------------------------------------------------------------------------
OBJECT                                  = COLUMN
NAME                                    = "C TIME"
UNIT                                    = "SECONDS"
DATA_TYPE                               = REAL
START_BYTE                              = 1
BYTES                                   = 8
MISSING                                 = 1.0E+32
DESCRIPTION                             = "
Time column. This field contains time in seconds after Jan 01, 1966 but is displayed in the default time format selected by the user."
END_OBJECT                              = COLUMN

OBJECT                                  = COLUMN
NAME                                    = "PDS TIME"
UNIT                                    = "TIME"
DATA_TYPE                               = CHARACTER
 START_BYTE                             = 9
BYTES                                   = 24
DESCRIPTION                             = "
Date/Time string of the form yyyy-mm-ddThh:mm:ss.sss such that the representation of the date Jan 01, 2000  00:00:00.000 would
be 2000-01-01T00:00:00.000Z (Z indicates Universal Time)."
END_OBJECT                              = COLUMN

OBJECT                                  = COLUMN
NAME                                    = "D1 RATE"
UNIT                                    = "COUNTS"
DATA_TYPE                               = "REAL"
START_BYTE                              = 33
BYTES                                   = 4
MISSING                                 = 1.0E+32
DESCRIPTION                             = "
The D1 rate is approximately porportional to the omnidirectional flux of electrons with kinetic energy > ~1MeV. To obtain greater
accuracy, the D1 calibration tables (see catalog) should be applied."
END_OBJECT                              = COLUMN
```

A.27.3 TABLE Variations

This section addresses a number of structural variations of “table based” data objects. As the structure of SERIES and SPECTRUM objects are similar and can be identical to the TABLE object, all three objects (TABLE, SERIES, and SPECTRUM) can be of the following structure types. The structural variations presented here are primarily due to the physical placement of the data (ROW_BYTES) in relation to the size of the data record (RECORD_BYTES), the type of the data (ASCII or BINARY), and the format of the data (FIXED_LENGTH or STREAM).

This section is not intended to be a complete reference for TABLE variations. Within the following examples, some illustrate a recommended data modelling approach, some illustrate alternate approaches, and other examples are included solely to document their existence.

Note: The examples in the following sections use OBJECT = TABLE, but OBJECT = SERIES or OBJECT = SPECTRUM could be substituted.

A.27.3.1 Record blocking in Fixed Length TABLES

The PDS recommended TABLE format requires the ROW_BYTES of the TABLE object to be equal to RECORD_BYTES of the file. This is not always the case, particularly when describing existing binary TABLE formats.

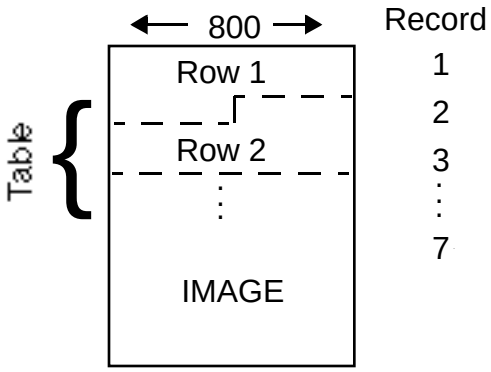
A common use of blocking occurs when two or more data objects are packaged into the same file, each requiring a different size record. In addition, rows in a TABLE are sometimes blocked into larger physical records to minimize input/output operations.

Rows in both ASCII or binary tables can be either larger or smaller than the physical record size specified by the RECORD_BYTES keyword.

Example - Binary Table with ROW_BYTES > RECORD_BYTES

The following label subset and illustration provide the general characteristics of a product containing an 800 byte IMAGE object together with a TABLE with 1200 byte rows:

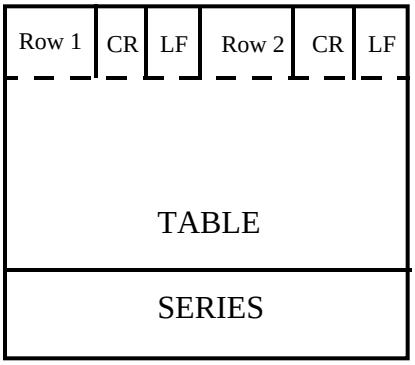
```
RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 800
^TABLE                         =("IMAGE.IMG",1)
^IMAGE                         =("IMAGE.IMG",7)
  ...
OBJECT                         = TABLE
INTERCHANGE_FORMAT             = BINARY
ROW_BYTES                      = 1200
   ...
END_OBJECT                     = TABLE

OBJECT                         = IMAGE
SAMPLES                        = 800
SAMPLE_BITS                    = 8
   ...
END_OBJECT                     = IMAGE
```



Example - ASCII Table with ROW_BYTES < RECORD_BYTES

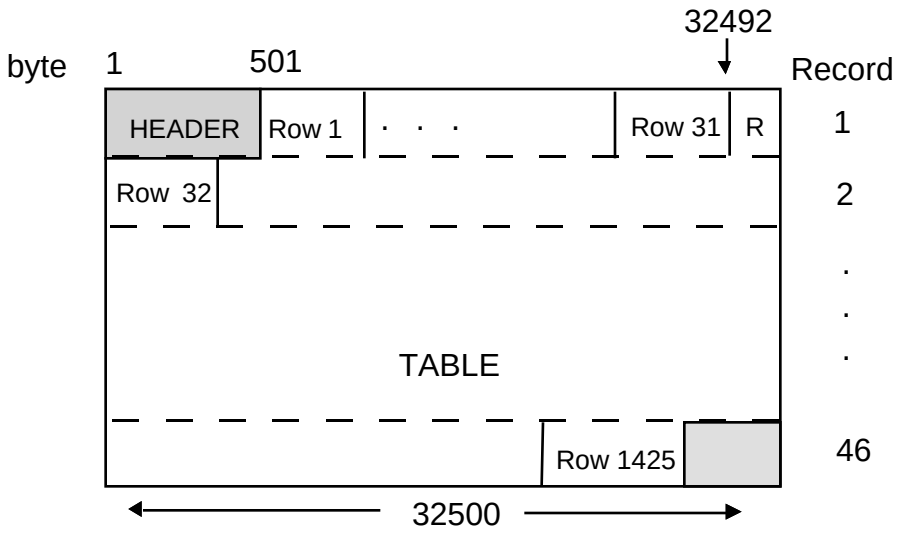
The following label subset and illustration provide the general characteristics of a product containing a SERIES object containing 800 byte rows together with a TABLE object with 400 byte rows:

```
RECORD_TYPE                = FIXED_LENGTH
RECORD_BYTES               = 800
...

OBJECT                     = TABLE
INTERCHANGE_FORMAT         = ASCII
ROW_BYTES                  =400
...
END_OBJECT                 = TABLE

OBJECT                     =SERIES
INTERCHANGE_FORMAT         =ASCII
ROW_BYTES                  = 800
...
END_OBJECT                 = SERIES
```



Example - Binary Table with ROW_BYTES < RECORD_BYTES

The following label subset and illustration provide the general characteristics of a product containing an HEADER object containing one 500 byte row together with a TABLE with 1032 byte rows. In this case, both the HEADER and TABLE rows are blocked into 32500 byte records. Note that the rows cross record boundaries.

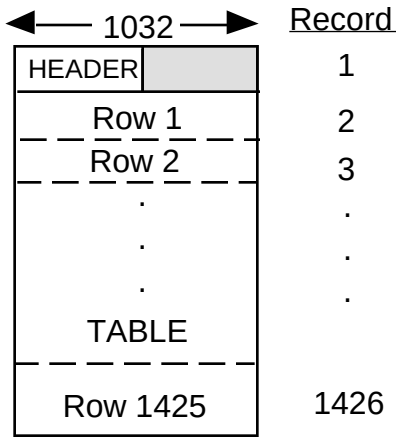


```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3

/* FILE CHARACTERISTICS */
RECORD_TYPE                             = FIXED_LENGTH
RECORD_BYTES                            = 32500
FILE_RECORDS                            = 46
^HEADER                                 = ("ADF01141.3",1)
^TABLE                                  = ("ADF01141.3",501<BYTES>)

/* IDENTIFICATION KEYWORDS */
DATA_SET_ID                             = "MGN-V-RDRS-5-CDR-ALT/RAD-V1.0"
PRODUCT_ID                              = "ADF01141.3"
TARGET_NAME                             = VENUS
SPACECRAFT_NAME                         = MAGELLAN
INSTRUMENT_NAME                         = "RADAR SYSTEM"
MISSION_PHASE_NAME                      = PRIMARY_MISSION
PRODUCT_CREATION_TIME                   = 1991-07-23T06:16:02.000Z
ORBIT_NUMBER                            = 1141
START_TIME                              = UNK
STOP_TIME                               = UNK
SPACECRAFT_CLOCK_START_COUNT            = UNK
SPACECRAFT_CLOCK_STOP_COUNT             = UNK
HARDWARE_VERSION_ID                     = 01
SOFTWARE_VERSION_ID                     = 02
UPLOAD_ID                               = M0356N
NAVIGATION_SOLUTION_ID                  = "ID = M0361-12 "
DESCRIPTION                             = " This file contains binary records describing, in time order, each altimeter
footprint measured during an orbit of the Magellan radar mapper."

/* DATA OBJECT DEFINITION  DESCRIPTIONS */
OBJECT                                  = HEADER
HEADER_TYPE                             = SFDU
BYTES                                   = 500
END_OBJECT                                  = HEADER
OBJECT                                  = TABLE
INTERCHANGE_FORMAT                      = BINARY
ROWS                                    = 1425
COLUMNS                                 = 40
ROW_BYTES                               = 1032
^STRUCTURE                              = "ADFTBL.FMT"
END_OBJECT                                      = TABLE
END

Contents of format file "ADFTBL.FMT"
--------------------------------------
OBJECT                                  = COLUMN
NAME                                    = SFDU_LABEL_AND_LENGTH
START_BYTE                              = 1
DATA_TYPE                               = CHARACTER
BYTES                                   = 20
UNIT                                    = "N/A"
DESCRIPTION                             = "
The SFDU_label_and_length element identifies the label and length of the Standard Format Data Unit (SFDU)."
END_OBJECT                              = COLUMN

OBJECT                                  = COLUMN
NAME                                    = FOOTPRINT_NUMBER
START_BYTE                              = 21
```

```
DATA_TYPE                       = LSB_INTEGER
BYTES                           = 4
UNIT                            = "N/A"
DESCRIPTION                     = "The footprint_number element provides a signed integer value. The altimetry
and radiometry processing program assigns footprint 0 to that observed at nadir at periapsis. The remaining footprints are located
along the spacecraft nadir track, with a separation that depends on the Doppler resolution of the altimeter at the epoch at which that
footprint is observed. Pre-periapsis footprints will be assigned negative numbers, post-periapsis footprints will be assigned positive
ones. A loss of several consecutive burst records from the ALT-EDR will result in missing footprint numbers."

END_OBJECT                      = COLUMN

...

OBJECT                          = COLUMN
 NAME                           = DERIVED_THRESH_DETECTOR_INDEX
 START_BYTE                     = 1001
 DATA_TYPE                      = LSB_UNSIGNED_INTEGER
 BYTES                          = 4
 UNIT                           = "N/A"
 DESCRIPTION                    = "The derived_thresh_detector_index element provides the value of the element
in range_sharp_echo_profile that satisfies the altimeter threshold detection algorithm, representing the distance to the nearest object
in this radar footprint in units of 33.2 meters, modulus a 10.02 kilometer altimeter range ambiguity."
END_OBJECT                      = COLUMN
```

Example - Alternate format; PDS Recommended

The following label subset and illustration provide an alternate data organization for the preceding example. In this example, a record size of 1032 is used to match the row size of the TABLE, and the 500 byte HEADER uses only a portion of the first record. This organization would conform to the PDS recommended TABLE structure.

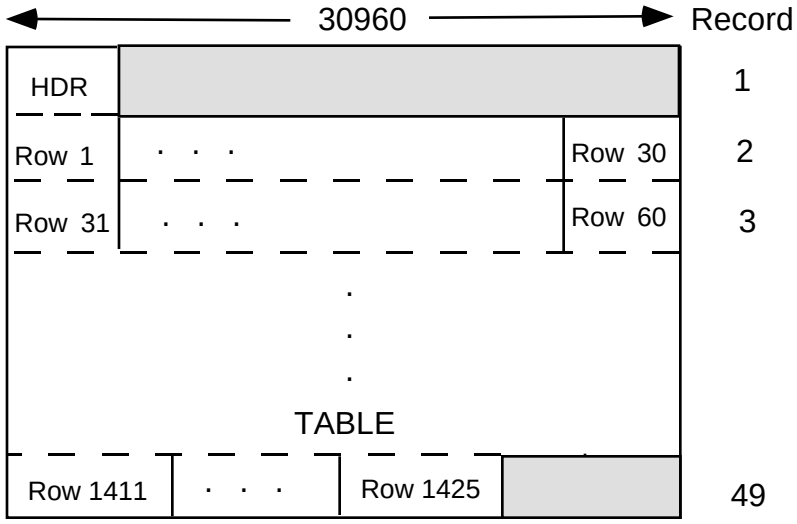



```
...
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 1032
FILE_RECORDS                    = 1426
^HEADER                         = ("ADF01141.3",1)
^TABLE                          = ("ADF01141.3",2)
```

```
 ...

/* DATA OBJECT DEFINITIONS */
OBJECT                          = HEADER
 HEADER_TYPE                    = SFDU
 BYTES                          = 500
END_OBJECT
OBJECT                          = TABLE
 INTERCHANGE_FORMAT             = BINARY
 ROWS                           = 1425
 COLUMNS                        = 40
 ROW_BYTES                      = 1032
 ^STRUCTURE                     = "ADFTBL.FMT"
END_OBJECT
END
```

Example - Alternate format; Rows on Record Boundaries

The following label subset and illustration provide a second alternate data organization for the preceding example. In this example, a record size of 66048 is used to hold 30 rows of the TABLE. Again the 500 byte HEADER uses only a portion of the first record.




```
...
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    =30960
FILE_RECORDS                    =49
^HEADER                         = ("ADF01141.3",1)
^TABLE                          = ("ADF01141.3",2)
 ...

/* DATA OBJECT DEFINITIONS */
OBJECT                          = HEADER
 HEADER_TYPE                    = SFDU
 BYTES                          = 500
END_OBJECT
```

```
OBJECT                          = TABLE
 INTERCHANGE_FORMAT             = BINARY
 ROWS                           = 1425
 COLUMNS                        = 40
 ROW_BYTES                      = 1032
 ^STRUCTURE                     = "ADFTBL.FMT"
END_OBJECT
END
```

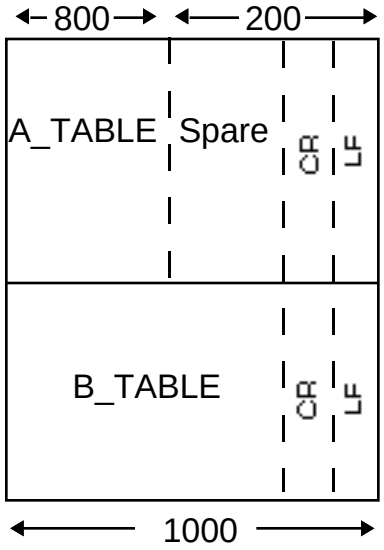
A.27.3.2 Multiple TABLEs with varying ROW_BYTES

A data product may contain several ASCII or binary tables, each with a different row size.

Example - Fixed Length Records - Multiple ASCII tables

The following label subset and illustration utilizes fixed length records of the maximum row size. The smaller table is padded with spares preceding the <CR><LF>. Note that the ROW_BYTES keyword in A_TABLE could be replaced by ROW_BYTES = 800 and ROW_SUFFIX_BYTES = 200. See section A.27.4 for further information on handling spares.

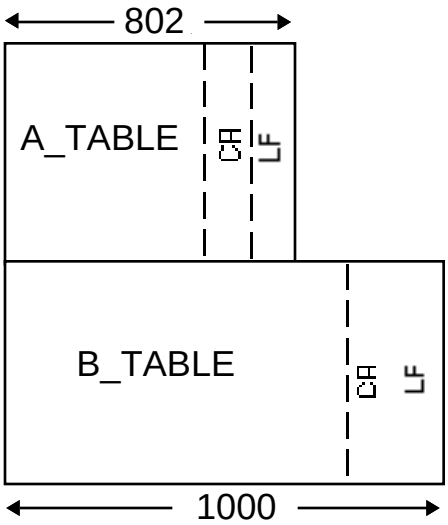
```
RECORD_TYPE                     = FIXED_LENGTH
RECORD_BYTES                    = 1000
   ...
OBJECT                          = A_TABLE
INTERCHANGE_FORMAT              = ASCII
ROW_BYTES                       = 1000
     ...
END_OBJECT                      = A_TABLE

OBJECT                          = B_TABLE
INTERCHANGE_FORMAT              = ASCII
ROW_BYTES                       = 1000
     ...
END_OBJECT                      = B_TABLE
```



Example - Stream Records - Multiple ASCII tables

The following label subset and illustration utilizes stream records for the same data as the previous example, placing the <CR><LF> pair at the end of the data in each table. There is no need to pad out the smaller table using the STREAM format, and the RECORD_BYTES keyword is not applicable.

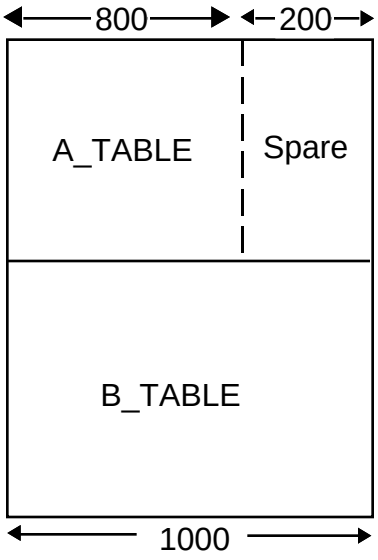
```
RECORD_TYPE                    = STREAM
   ...
OBJECT                         = A_TABLE
INTERCHANGE_FORMAT             = ASCII
ROW_BYTES                      = 802
   ...
END_OBJECT                     = A_TABLE

OBJECT                         = B_TABLE
INTERCHANGE_FORMAT             = ASCII
ROW_BYTES                      = 1000
   ...
   END_OBJECT = B_TABLE
```



Example - Fixed Length Records - Multiple Binary tables

The following label subset and illustration utilizes fixed length records of the maximum row size. The smaller table has a spare set of bytes in each record, explicitly defined in a “spare” COLUMN object. Note that the ROW_BYTES keyword in A_TABLE could be replaced by ROW_BYTES = 800 and ROW_SUFFIX_BYTES = 200, instead of explicitly defining the SPARE column. See section A.27.4 for further information on handling spares.

```
RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 1000
   ...
OBJECT                         = A_TABLE
INTERCHANGE_FORMAT             = BINARY
ROW_BYTES                      = 1000
    ...
OBJECT                         = COLUMN
NAME                           = "SPARE"
DATA_TYPE                      = "N/A"
START_BYTE                     = 801
BYTES                          = 200
END_OBJECT                     = COLUMN
END_OBJECT                     = A_TABLE

OBJECT                         = B_TABLE
INTERCHANGE_FORMAT             = BINARY
ROW_BYTES                      = 1000
   ...
END_OBJECT                     = B_TABLE
```



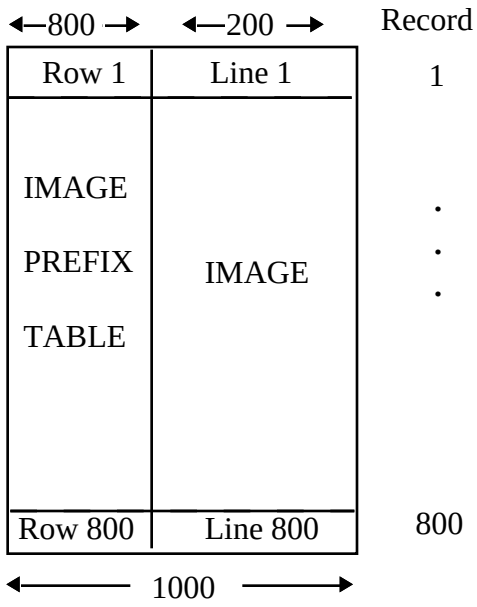
A.27.3.3 ROW_PREFIX or ROW_SUFFIX use

There are currently two methods to utilize ROW_PREFIX_BYTES and ROW_SUFFIX_BYTES in TABLE objects. The first application is limited to Binary TABLE objects that are adjacent to another object, such that each object shares the same record in a file. The second application is for identifying spare bytes at the beginning or end of a record that are not considered part of the TABLE data.

Example - Row Suffix use for compound TABLE and IMAGE

The following label subset and illustration utilizes fixed length records each containing a row of a TABLE data object, and a line of an IMAGE object. This is a common format for providing ancillary information applicable to each IMAGE line.

```
RECORD_TYPE                = FIXED_LENGTH
RECORD_BYTES               = 1000
   ...
OBJECT                     = TABLE
INTERCHANGE_FORMAT         = BINARY
ROW_BYTES                  = 200
ROW_SUFFIX_BYTES           = 800
   ...
END_OBJECT                 = TABLE

OBJECT                     = IMAGE
LINE_SAMPLES               = 800
SAMPLE_BITS                = 8
LINE_PREFIX_BYTES          = 200
   ...
END_OBJECT                 = IMAGE
```



The following RULES apply to the use of ROW_PREFIX_BYTES and ROW_SUFFIX_BYTES:

1. For compound “table based” objects (TABLE, SPECTRUM, SERIES) in a data product, or for identifying Spare parts of a record:

RECORD_BYTES = ROW_BYTES + ROW_PREFIX_BYTES + ROW_SUFFIX_BYTES

2. For compound “table based” and IMAGE objects in a data product:

RECORD_BYTES = (LINE_SAMPLES * SAMPLE_BITS / 8) + ROW_PREFIX_BYTES + ROW_SUFFIX_BYTES

A.27.3.4 CONTAINER Object use

Complicated or lengthy tables that have a set of COLUMNS that repeat are often easier to describe with an illustration and the use of the CONTAINER sub-object in a TABLE description. The use of the container sub-object eliminates the need for repeating a group of COLUMN objects and adjusting the START_BYTE locations and descriptions for each repetition. Section A.8 provides an example of a TABLE utilizing the CONTAINER sub-object.

A.27.4 Guidelines for SPARE fields

There is often a need to reserve SPARE (or pad, filler, etc.). bytes in TABLE, SPECTRUM, and SERIES objects. While this is not required, it facilitates validation and ensures that the data producer did not inadvertently forget to account for some fields in the data. These guidelines differ slightly for BINARY and ASCII tables and FIXED_LENGTH or STREAM record files.

In all of the following guidelines, “embedded spares” refer to empty or spare bytes that are currently unused and are not defined as part of a data COLUMN.

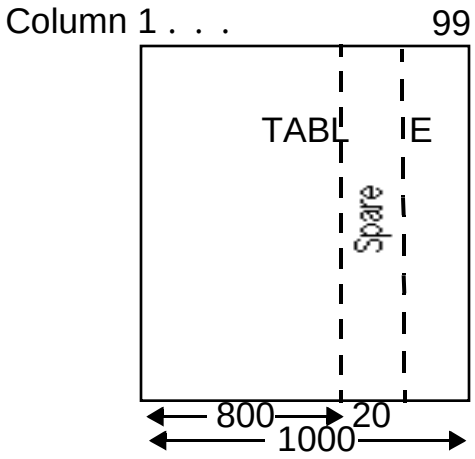
A.27.4.1 BINARY Tables - Fixed Length Records

The guidelines for handling SPARE fields in Fixed Length Binary Tables are:

- Embedded spares are allowed.
- Embedded spares are explicitly defined (with COLUMN Objects).
- Multiple Spare columns may all have NAME = SPARE
- Spares are allowed at the beginning or end of each row of data.
- Spares at the beginning or end of the data can be identified with
 1) an explicit COLUMN object or
or
 2) use of ROW_PREFIX_BYTES or ROW_SUFFIX_BYTES (note that these bytes should not be included in the value of ROW_BYTES)
- DATA_TYPE for Spare COLUMNS in binary table is 'N/A'

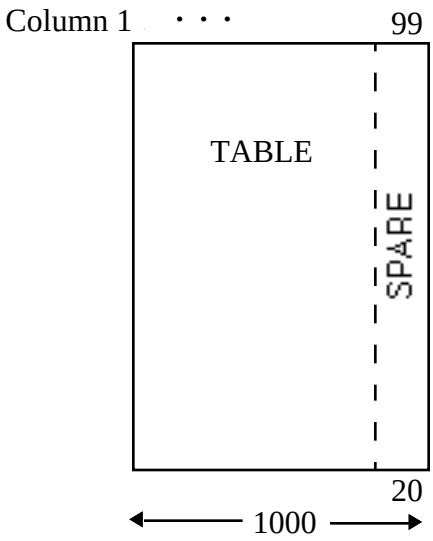
Example - SPARE field embedded in a Binary TABLE

```
_____________________________________________
RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 1000
   ...
OBJECT                         = TABLE
INTERCHANGE_FORMAT             = BINARY
ROW_BYTES                      = 1000
COLUMNS                        = 99
     ...
OBJECT                         = COLUMN
NAME                           = SPARE
START_BYTE                     = 801
BYTES                          = 20
DATA_TYPE                      = "N/A"
      ...
END_OBJECT                     = COLUMN
END_OBJECT                     = TABLE
```



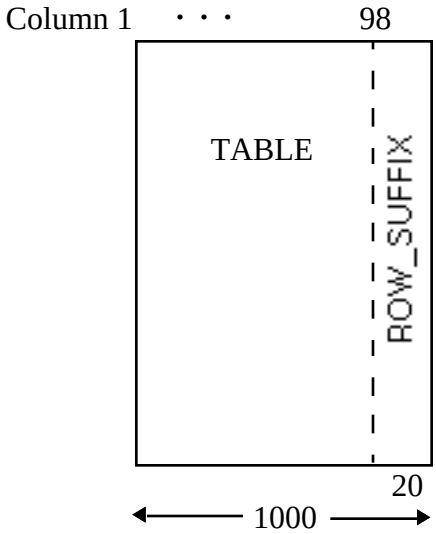
Example - Spares at end of a Binary TABLE - Explicit 'SPARE' Column

```
RECORD_TYPE                = FIXED_LENGTH
RECORD_BYTES               = 1000
   ...
OBJECT                     = TABLE
INTERCHANGE_FORMAT         = BINARY
ROW_BYTES                  = 1000
COLUMNS                    = 99
     ...
OBJECT                     = COLUMN
NAME                       = SPARE
BYTES                      = 20
DATA_TYPE                  = "N/A"
START_BYTE                 = 981
       ...
END_OBJECT                 = COLUMN
END_OBJECT                 = TABLE
```



Example - Spares at end of a Binary TABLE - ROW_SUFFIX use

```
RECORD_TYPE                = FIXED_LENGTH
RECORD_BYTES               = 1000
   ...
OBJECT                     = TABLE
INTERCHANGE_FORMAT         = BINARY
ROW_BYTES                  = 980
ROW_SUFFIX_BYTES           = 20
COLUMNS                    = 98
   ...
END_OBJECT                 = TABLE
```



A.27.4.2 ASCII Tables - Fixed Length Records

In ASCII tables, field delimiters (") and (,) and the <CR><LF> pair are considered part of the data, even though the COLUMN objects attributes do not include them. Spares in ASCII tables are limited to the “space” character (ASCII 20). The guidelines for handling SPARE fields in Fixed Length ASCII Tables are:

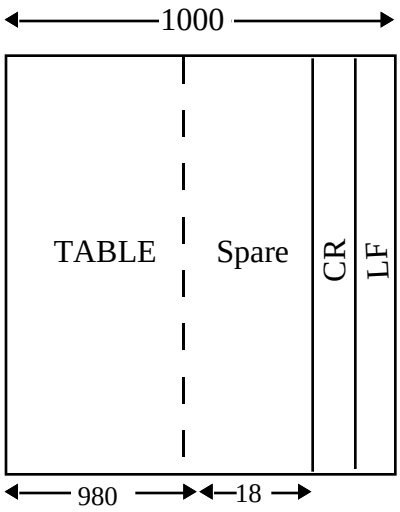
- Embedded spares are not allowed.
- Spares are allowed at the end of each row of data.
- The <CR><LF> follows the spare data.
- There are no delimiters (commas or quotes) surrounding the spares.
- Spares at the end of the data can be ignored (like field delimiters and CR LF) or they can be identified

1) in the Table Description
or
2) by using ROW_SUFFIX_BYTES (note that these bytes should not be included in the value of ROW_BYTES)

Example - SPARE field at end of ASCII TABLE - Table description note

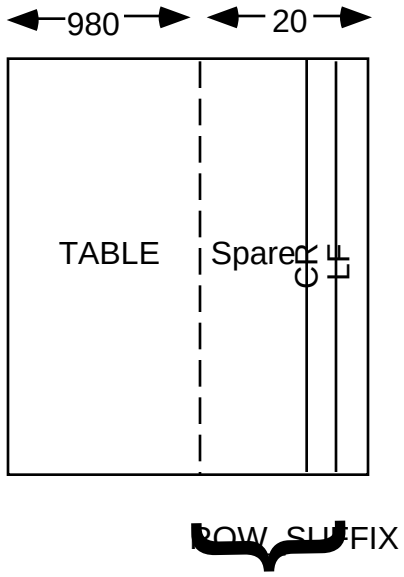
---

```
RECORD_TYPE                    = FIXED_LENGTH
RECORD_BYTES                   = 1000
...
OBJECT                         = TABLE
INTERCHANGE_FORMAT             = ASCII
ROW_BYTES                      = 1000
...

DECRIPTION                     ="This table contains
980 bytes of table data followed by 18 bytes of blank spares.
Byte 999 and 1000 contain the <CR> <LF>
pair."
```



Example - Spares at end of a ASCII TABLE - ROW_SUFFIX use.

```
RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 1000
...
OBJECT                      = TABLE
INTERCHANGE_FORMAT          = ASCII
ROW_BYTES                   = 980
ROW_SUFFIX_BYTES            = 20
...
END_OBJECT                  = TABLE
```



A.27.5 ASCII Tables - STREAM Records

Spares are not used with ASCII Tables in STREAM record formats. In STREAM files, the last data field explicitly defined with a COLUMN object is followed immediately by the <CR><LF> pair. Since there is no use for spares at the end of the data, and embedded spares are not allowed in ASCII tables, spares are not applicable here.

A.28 TEXT

The TEXT object contains plain text which begins immediately after the END statement. It is recommended that TEXT objects contain no special formatting characters, with the exception of the carriage return/line feed sequence and the page break. Tabs are discouraged, since they are interpreted differently by different programs. It is important to include BOTH the carriage return and line feed characters when preparing files for use on a variety of host systems.

Use of the Macintosh or Unix line terminators will cause text to be unreadable on other host computers. It is recommended that text lines be limited to 80 characters inclusive of the Carriage Return (Control M, HexOxOd) and Line Feed (Control J, HexOxOa) line delimiters.

NOTE: The text object is used in files describing the contents of an archive volume or the contents of a directory, such as AAREADME.TXT, DOCINFO.TXT, VOLINFO.TXT, SOFTINFO.TXT, etc. These files must be in plain unmarked ASCII text and always have the file name extension of .TXT. Documents placed on the volume in plain ASCII text, on the other hand, must be described using the DOCUMENT object. (See the definition of the DOCUMENT Object in Appendix A.)

The NOTE field provides a brief introduction to the TEXT.

Required Keywords

1. NOTE
2. PUBLICATION_DATE

Optional Keywords

1. INTERCHANGE_FORMAT

Required Objects

None

Optional Objects

None

Example

The example below is a portion of an AAREADME.TXT file.

```
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3
RECORD_TYPE                             = STREAM

OBJECT                                  = TEXT
PUBLICATION_DATE                        = 1991-05-28
NOTE                                    = "Introduction to this CD-ROM volume."
END_OBJECT                              = TEXT
END
```

GEOLOGIC REMOTE SENSING FIELD EXPERIMENT

This set of compact read-only optical disks (CD-ROMs) contains a data collection acquired by ground-based and airborne instruments during the Geologic Remote Sensing Field Experiment (GRSFE). Extensive documentation is also included. GRSFE took place in July, September, and October, 1989, in the southern Mojave Desert, Death Valley, and the Lunar Crater Volcanic Field, Nevada. The purpose of these CD-ROMs is to make available in a compact form through the Planetary Data System (PDS) a collection of relevant data to conduct analyses in preparation for the Earth Observing System (EOS), Mars Observer (MO), and other missions. The generation of this set of CD-ROMs was sponsored by the NASA Planetary Geology and Geophysics Program, the Planetary Data System (PDS) and the Pilot Land Data System (PLDS).

This AAREADME.TXT file is one of the two nondirectory files located in the top level directory of each CD-ROM volume in this collection. The other file, VOLDESC.CAT, contains an overview of the data sets on these CD-ROMs and is written in a format that is designed for access by computers. These two files appear on every volume in the collection. All other files on the CD-ROMs are located in directories below the top level directory ....

A.29 VOLUME

The VOLUME object describes a physical or logical unit used to store or distribute data products (e.g. a magnetic tape, CD-ROM disk, On-Line Magnetic disk or floppy disk) which contain directories and files. The directories and files may include documentation, software, calibration and geometry information as well as the actual science data.

Required Keywords

1. DATA_SET_ID
2. DESCRIPTION
3. MEDIUM_TYPE
4. PUBLICATION_DATE
5. VOLUME_FORMAT
6. VOLUME_ID
7. VOLUME_NAME
8. VOLUME_SERIES_NAME
9. VOLUME_SET_NAME
10. VOLUME_SET_ID
11. VOLUME_VERSION_ID
12. VOLUMES

Optional Keywords

1. BLOCK_BYTES
2. DATA_SET_COLLECTION_ID
3. FILES
4. HARDWARE_MODEL_ID
5. LOGICAL_VOLUMES
6. LOGICAL_VOLUME_PATH_NAME
7. MEDIUM_FORMAT
8. NOTE
9. OPERATING_SYSTEM_ID
10. PRODUCT_TYPE
11. TRANSFER_COMMAND_TEXT
12. VOLUME_INSERT_TEXT

Required Objects

1. CATALOG
2. DATA_PRODUCER

Optional Objects

1. DIRECTORY
2. FILE
3. DATA_SUPPLIER

Example 1 (Typical CD-ROM Volume)

Please see example in A.5 CATALOG.

Example 2 (Tape Volume)

The following VOLUME object example shows how directories and files are indicated when a volume is stored on ANSI tape for transfer. This form should be used when transferring volumes of data on media which do not support hierarchical directory structures (for example, submitting a volume of data for premastering). The VOLDESC.CAT file will contain the standard volume keywords, but the values of MEDIUM_TYPE, MEDIUM_FORMAT and VOLUME_FORMAT indicate that the volume is stored on tape.

In this example two files are defined in the root directory of the volume, VOLDESC.CAT and AAREADME.TXT. The first directory object defines the CATALOG directory which contains meta data in the High Level Catalog Templates. Here they all exist in one file, CATALOG.CAT. The second directory object defines an INDEX subdirectory, with three files embedded in it (INDXINFO.TXT, INDEX.LBL, INDEX.TAB). Following that directory, the first data directory is defined. Note that the sequence number field indicates the sequence of the file on the tape volume.

```
------------------------------------------------------------------------
CCSD3ZF0000100000001NJPL3IF0PDSX00000001
PDS_VERSION_ID                          = PDS3
OBJECT                                  = VOLUME
VOLUME_SERIES_NAME                      = "MISSION TO MARS"
VOLUME_SET_NAME                         = "MARS DIGITAL IMAGE MOSAIC AND DIGITAL TERRAIN MODEL"
VOLUME_SET_ID                           = USA_NASA_PDS_VO_2001_TO_VO_2007
VOLUMES                                 = 7
VOLUME_NAME                             = "MDIM/DTM VOLUME 7: GLOBAL COVERAGE"
VOLUME_ID                               = VO_2007
VOLUME_VERSION_ID                       = "VERSION 1"
PUBLICATION_DATE                        = 1992-04-01
DATA_SET_ID                             = "VO1/VO2-M-VIS-5-DTM-V1.0"
MEDIUM_TYPE                             = "8-MM HELICAL SCAN TAPE"
MEDIUM_FORMAT                           = "2 GB"
VOLUME_FORMAT                           = ANSI
HARDWARE_MODEL_ID                       = "VAX 11/750"
OPERATING_SYSTEM_ID                     = "VMS 4.6"
DESCRIPTION                             = "This volume contains the Mars Digital Terrain Model and Mosaicked Digital
Image Model covering the entire planet at resolutions of 1/64 and 1/16 degree/pixel.  The volume also contains Polar Stereographic
projection files of the north and south pole areas from 80 to 90 degrees latitude; Mars Shaded Relief Airbrush Maps at 1/16 and 1/
4 degree/pixel; a gazetteer of Mars features; and a table of updated viewing geometry files of the Viking EDR images that comprise
the MDIM."
MISSION_NAME                            = VIKING
SPACECRAFT_NAME                         = {VIKING_ORBITER_1,VIKING_ORBITER_2}
SPACECRAFT_ID                           = {VO1,VO2}
```

```
OBJECT                          = DATA_PRODUCER
INSTITUTION_NAME                = "U.S.G.S. FLAGSTAFF"
FACILITY_NAME                   = "BRANCH OF ASTROGEOLOGY"
FULL_NAME                       = "Eric M. Eliason"
DISCIPLINE_NAME                 = "IMAGE PROCESSING"
ADDRESS_TEXT                    = " Branch of Astrogeology \n
                                  United States Geological Survey\n
                                  2255 North Gemini Drive\n
                                   Flagstaff, Arizona. 86001 USA"
END_OBJECT                      = DATA_PRODUCER

OBJECT                          = CATALOG
^CATALOG                        = "CATALOG.CAT"
END_OBJECT                      = CATALOG

OBJECT                          = FILE
FILE_NAME                       = "VOLDESC.CAT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 1
END_OBJECT                      = FILE

OBJECT                          = FILE
FILE_NAME                       = "AAREADME.TXT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 2
END_OBJECT                      = FILE

OBJECT                          = DIRECTORY
NAME                            = CATALOG

OBJECT                          = FILE
FILE_NAME                       = "CATALOG.CAT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 3
END_OBJECT                      = FILE
END_OBJECT                      = DIRECTORY

OBJECT                          = DIRECTORY
NAME                            = DOCUMENT

OBJECT                          = FILE
FILE_NAME                       = "VOLINFO.TXT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 4
END_OBJECT                      = FILE

OBJECT                          = FILE
FILE_NAME                       = "DOCINFO.TXT"
RECORD_TYPE                     = STREAM
SEQUENCE_NUMBER                 = 5
END_OBJECT                      = FILE
END_OBJECT                      = DIRECTORY

OBJECT                          = DIRECTORY
NAME                            = INDEX

OBJECT                          = FILE
FILE_NAME                       = "INDXINFO.TXT"
```

```
RECORD_TYPE                 = STREAM
SEQUENCE_NUMBER             = 6
END_OBJECT                  = FILE

OBJECT                      = FILE
FILE_NAME                   = "INDEX.LBL"
RECORD_TYPE                 = STREAM
SEQUENCE_NUMBER             = 7
END_OBJECT                  = FILE

OBJECT                      = FILE
FILE_NAME                   = "INDEX.TAB"
RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 512
FILE_RECORDS                = 6822
SEQUENCE_NUMBER             = 8
END_OBJECT                  = FILE
END_OBJECT                  = DIRECTORY

OBJECT                      = DIRECTORY
NAME                        = MG00NXXX

OBJECT                      = FILE
FILE_NAME                   = "MG00N007.IMG"
RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 964
FILE_RECORDS                = 965
SEQUENCE_NUMBER             = 9
END_OBJECT                  = FILE

OBJECT                      = FILE
FILE_NAME                   = "MG00N012.IMG"
RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 964
FILE_RECORDS                = 965
SEQUENCE_NUMBER             = 10
END_OBJECT                  = FILE

END_OBJECT                  = DIRECTORY

END_OBJECT                  = VOLUME
END
```

Example 3a (CD-ROM Volume containing logical volumes)
Examples 3a and 3b illustrate the use of the VOLUME Object in the top level and at the logical volume level of an archive volume. Note that the VOLUME Object is required at both levels.

For examples 3a and 3b, the CD-ROM is structured as three separate logical volumes with root directories named PPS/, UVS/ and RSS/. An additional SOFTWARE directory is supplied at volume root for use with all logical volumes.

Example 3a illustrates the use of the VOLUME Object present at the top level of a CD-ROM containing logical volumes. Note usage of the keywords DATA_SET_ID, LOGICAL_VOLUMES, and LOGICAL_VOLUME_PATH_NAME.

```
PDS_VERSION_ID                  = PDS3
OBJECT = VOLUME
VOLUME_SERIES_NAME              = "VOYAGERS TO THE OUTER PLANETS"
VOLUME_SET_NAME                 = "PLANETARY RING OCCULTATIONS FROM VOYAGER"
VOLUME_SET_ID                   = "USA_NASA_PDS_VG_3001"
VOLUMES                         = 1
MEDIUM_TYPE                     = "CD-ROM"
VOLUME_FORMAT                   = "ISO-9660"
VOLUME_NAME                     = "VOYAGER PPS/UVS/RSS RING OCCULTATIONS"
VOLUME_ID                       = "VG_3001"
VOLUME_VERSION_ID               = "VERSION 1"
PUBLICATION_DATE                = 1994-03-01
DATA_SET_ID                     = {"VG2-SR/UR/NR-PPS-4-OCC-V1.0",
                                  "VG1/VG2-SR/UR/NR-UVS-4-OCC-V1.0","VG1/VG2-SR/UR/NR-RSS-4-
                                  OCC-V1.0"}
LOGICAL_VOLUMES                 = 3
LOGICAL_VOLUME_PATH_NAME        = {"PPS/", "UVS/", "RSS/"}
DESCRIPTION                     = "This volume contains the Voyager 1 and Voyager 2 PPS/UVS/RSS ring
occultation and ODR data sets. Included are data files at a variety of levels of processing, plus ancillary geometry, calibration and
trajectory files plus software and documentation.

This CD-ROM is structured as three separate logical volumes with root directories named PPS/, UVS/ and RSS/. An additional
SOFTWARE directory is supplied at volume root for use with all logical volumes."

OBJECT                          = DATA_PRODUCER
INSTITUTION_NAME                = "PDS RINGS NODE"
FACILITY_NAME                   = "NASA AMES RESEARCH CENTER"
FULL_NAME                       = "DR. MARK R. SHOWALTER"
DISCIPLINE_NAME                 = "RINGS"
ADDRESS_TEXT                    = "Mail Stop 245-3 \n
                                   NASA Ames Research Center \n
                                   Moffett Field, CA 94035-1000"
END_OBJECT                      = DATA_PRODUCER

OBJECT                          = CATALOG
DATA_SET_ID                     = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
LOGICAL_VOLUME_PATH_NAME        = "PPS/"
^MISSION_CATALOG                = "MISSION.CAT"
^INSTRUMENT_HOST_CATALOG        = "INSTHOST.CAT"
^INSTRUMENT_CATALOG             = "INST.CAT"
^DATA_SET_COLLECTION_CATALOG    = "DSCOLL.CAT"
^DATA_SET_CATALOG               = "DATASET.CAT"
^REFERENCE_CATALOG               = "REF.CAT"
^PERSONNEL_CATALOG              = "PERSON.CAT"
```

```
END_OBJECT                          = CATALOG

OBJECT                              = CATALOG
DATA_SET_ID                         = "VG1/VG2-SR/UR/NR-UVS-4-OCC-V1.0"
LOGICAL_VOLUME_PATH_NAME            = "UVS/"
^MISSION_CATALOG                    = "MISSION.CAT"
^INSTRUMENT_HOST_CATALOG            = "INSTHOST.CAT"
^INSTRUMENT_CATALOG                 = "INST.CAT"
^DATA_SET_COLLECTION_CATALOG        ="DSCOLL.CAT"
^DATA_SET_CATALOG                   = "DATASET.CAT"
^REFERENCE_CATALOG                  = "REF.CAT"
^PERSONNEL_CATALOG                  = "PERSON.CAT"
END_OBJECT                          = CATALOG

OBJECT                              = CATALOG
DATA_SET_ID                         = "VG1/VG2-SR/UR/NR-RSS-4-OCC-V1.0"
LOGICAL_VOLUME_PATH_NAME            = "RSS/"
^MISSION_CATALOG                    = "MISSION.CAT"
^INSTRUMENT_HOST_CATALOG            = "INSTHOST.CAT"
^INSTRUMENT_CATALOG                 = "INST.CAT"
^DATA_SET_COLLECTION_CATALOG        = "DSCOLL.CAT"
^DATA_SET_CATALOG                   = "DATASET.CAT"
^REFERENCE_CATALOG                  = "REF.CAT"
^PERSONNEL_CATALOG                  = "PERSON.CAT"
END_OBJECT                          = CATALOG

END_OBJECT                          = VOLUME
END
```

Example 3b (PPS/VOLDESC.CAT -- CD-ROM logical volume)

Example 3b illustrates the use of the Volume object which is required at the top level of a logical volume. Note the difference in values for the keywords DATA_SET_ID and LOGICAL_VOLUME_PATH_NAME from those used at the top level of the CD-ROM (example 3a). Also note that the keyword LOGICAL_VOLUMES does not appear here.

```
PDS_VERSION_ID                      = PDS3
OBJECT = VOLUME
VOLUME_SERIES_NAME                  = "VOYAGERS TO THE OUTER PLANETS"
VOLUME_SET_NAME                     = "PLANETARY RING OCCULTATIONS
                                    FROM VOYAGER"
VOLUME_SET_ID                       = "USA_NASA_PDS_VG_3001"
VOLUMES                             = 1
MEDIUM_TYPE                         = "CD-ROM"
VOLUME_FORMAT                       = "ISO-9660"
VOLUME_NAME                         = "VOYAGER PPS/UVS/RSS RING
                                    OCCULTATIONS"
VOLUME_ID                           = "VG_3001"
VOLUME_VERSION_ID                   = "VERSION 1"
PUBLICATION_DATE                    = 1994-03-01
DATA_SET_ID                         = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
LOGICAL_VOLUME_PATH_NAME            = "PPS/"
DESCRIPTION                         = "This logical volume contains the Voyager 2 PPS ring occultation data sets.
Included are data files at a variety of levels of processing, plus ancillary geometry, calibration  and trajectory files plus software and
documentation."

OBJECT                              = DATA_PRODUCER
```

```
INSTITUTION_NAME                = "PDS RINGS NODE"
FACILITY_NAME                   = "NASA AMES RESEARCH CENTER"
FULL_NAME                       = "DR. MARK R. SHOWALTER"
DISCIPLINE_NAME                 = "RINGS"
ADDRESS_TEXT                    = "Mail Stop 245-3
NASA Ames Research Center
Moffett Field, CA 94035-1000"
END_OBJECT                      = DATA_PRODUCER

OBJECT                          = CATALOG
DATA_SET_ID                     = "VG2-SR/UR/NR-PPS-4-OCC-V1.0"
LOGICAL_VOLUME_PATH_NAME        = "PPS/"
^MISSION_CATALOG                = "MISSION.CAT"
^INSTRUMENT_HOST_CATALOG        = "INSTHOST.CAT"
^INSTRUMENT_CATALOG             = "INST.CAT"
^DATA_SET_COLLECTION_CATALOG    = "DSCOLL.CAT"
^DATA_SET_CATALOG               = "DATASET.CAT"
^REFERENCE_CATALOG              = "REF.CAT"
^PERSONNEL_CATALOG              = "PERSON.CAT"
END_OBJECT                      = CATALOG

END_OBJECT                      = VOLUME
END
```